Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Škerjanc

Obogatena resničnost gibanja uporabnika v realnem času na mobilni napravi s pomočjo podatkov globinskega senzorja

magistrsko delo

študijski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana, 2016

Izjava o avtorstvu magistrskega dela

Spodaj podpisani Nejc Škerjanc sem avtor magistrskega dela z naslovom:

Obogatena resničnost gibanja uporabnika v realnem času na mobilni napravi s pomočjo podatkov globinskega senzorja

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Petra Peera,
- so elektronska oblika magistrskega dela, naslov (slovenski, angleški), povzetek (slovenski, angleški) ter ključne besede (slovenske, angleške) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, 20. novembra 2016 Podpis avtorja:

Za strokovno pomoč, usmerjanje in svetovanje pri izdelavi magistske naloge se zahvaljujem mentorju izr. prof. dr. Petru Peeru.

Posebna zahvala velja vsem domačim, ki so mi nudili podporo in me vztrajno spodbujali, da sem lahko dosegel uresničitev svojega cilja.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
UGC	user generated content	uporabniško kreirana vsebina
HOG	histogram of oriented gradients	histogram orientiranih gradientov
RGB	red & green & blue	rdeča & zelena & modra
FPS	frames per second	število slik na sekundo
SDK	software development kit	programska knjižnica
SLAM	simultaneous localization and mapping	hkratno določanje položaja in gradnja zemljevida
GPS	global positioning system	globalni sistem pozicioniranja
CAT	computer aided tomography	računalniško podprta tomografija

Povzetek

Naslov: Obogatena resničnost gibanja uporabnika v realnem času na mobilni napravi s pomočjo podatkov globinskega senzorja

Ljudje težimo k vedno bolj natančni rekonstrukciji realnega sveta. Uporaba samo RGB podatkovega toka ne daje zadovoljivih rezultatov pri detekciji in rekonstrukciji človeškega telesa. Z vpeljavo globinskega podatkovnega toka, ki nam ga posreduje globinski senzor, lahko rekonstruriramo okolico.

Algoritem z uporabo sinhronega RGB in globinskega podatkovnega toka izvaja detekcijo s prepletom obeh tokov. S poznavanjem fizikalnih zakonitosti človeškega telesa lahko detektiramo posamezne dele človeškega telesa. Na podlagi detekcije sledi izris animiranega lika (z uporabo slikovnega atlasa), ki kar najbolje prekriva detektiranega človeka in človeško telo.

Rezultat je knjižnica, napisana v C++ jeziku za čimboljšo prehodnost med platformami.

Kvalitativna in kvantitativna analiza je pokazala, da je z uporabo sinhronega RGB in globinskega podatkovnega toka detekcija človeškega telesa bolj natančna in učinkovita, kot samo uporaba posameznega podatkovnega toka.

Ključne besede: globinski senzor, detekcija okostja, animacija, obogatena resničnost, realni čas, mobilne naprave.

Abstract

Title: Real-time augmented reality of user motion on mobile device using depth sensor

People constantly aim to enhance the precision of the digital reconstruction of the real world. The detection and reconstruction of the human body does not provide comprehensive results with only RGB data stream. By introducing depth data flow from depth sensor we can create the reconstruction of our surroundings.

The algorithm performs the detection of the human body with the intertwine of synchronous RGB and depth data stream. With the knowledge of physical rules of the human body proportions we can detect human body parts. With the results of the detection the application then draws an animated character (using sprite sheet) that covers as much as possible of the previously detected human body.

The result is a library, which is written in the C++ programming language for a good transferability between platforms.

Qualitative and quantitative analysis show that using synchronous RGB and depth data stream make the reconstruction more accurate and effective than using only RGB or depth data stream.

Keywords: depth sensor, skeleton detection, animation, augmented reality, real-time, mobile devices.

Poglavje 1

Uvod

Računalniški vid je veda, ki interpretira slikovne podatke. Rezultat intepretacije je odvisen od namena in zadanih ciljev (detekcija, sledenje, klasifikacija, opis scene) [1]. Računalniški vid pokriva interpretacijo tako dvodimenzionalnih (2D) kot tudi tridimenzionalnih (3D) slik. Človeški možgani imajo dobro sposobnost interpretacije oblik in združevanja le-teh v celoto. V digitalnem zapisu se interpretacija izvaja s pomočjo algoritmov. Algoritmi zaznavajo sliko kot matriko števil. Za doseganje ciljne interpretacije se algoritmi poslužujejo matričnih manipulacij. Zato ima računalniški vid pomembno vlogo v industriji, prometu, medicini ipd [2].

Zadnji trendi računalniškega vida nakazujejo vse bolj sofisticirane sisteme, ki poleg vizualnih sistemov za pridobivanje podatkov uporabljajo tudi druge sisteme (GPS, pospeškomer ipd). Dodatne informacije pripomorejo k bolj učinkoviti in bolj natančni interpretaciji vizualnih podatkov. Primer takega sistema je samovozeči avtomobil podjetja Google [3].

Globinski senzorji so naprave, ki omogočajo prenašanja 3D informacije iz realnega sveta v virtualni prostor. Bliskovit razvoj so doživeli v zadnjih letih. Do leta 2011 so se globinski senzorji uporabljali predvsem v industriji, avtomobilizmu, vojski in znanosti. Leta 2011 je Microsoft predstavil barvno globinski vizualni sistem Kinect. Razvit je bil v sklopu izraelskega podjetja PrimeSense. Bila je prva naprava namenjena širši skupini uporabnikom in

učinkovitim SDK. SDK je omogočal enostaven dostop do globinskih podatkov in detekcijo človeških kretenj v realnem času.

V prihodnosti pričakujemo, da bodo postopoma vse mobilne naprave vsebovale globinske senzorje. Mobilnost v navezavi z globinskim senzorjem prinaša nove možnosti uporabe mobilnih naprav. Ena izmed možnosti uporabe tovrstne tehnologije je tudi industrija zabavnih aplikacij, katerih prvotni namen je zabava. Uporabniška izkušnja bo dosegla nov nivo (iskanje objektov, vzorcev, izvajanje kretenj ipd).

Naloga predstavlja tudi Structor senzor, ki je trenutno edini globinski senzor namenjen uporabi na mobilnih napravah. Ker se namesti kot zunanja razširitev, je predstavnik prehodne tehnologije globinskih senzorjev na mobilnih napravah – dokler globinski senzorji niso vgrajeni v mobilne naprave.

Cartoob [4] je serija aplikacij zabavne narave. Izvaja detekcijo človeškega telesa na RGB podatkovnem toku. Na podlagi detekcije čez uporabnika nariše enega izmed 27 predefiniranih poz animiranega lika. Zaradi uporabe samo RGB podatkovnega toka je uporabniška izkušnja slabša od željene. Z uporabo Structure senzorja poskušamo izboljšati uporabniško izkušnjo. Želimo doseči, da poze človeškega telesa ne bodo definirane vnaprej kot pri Cartoob. Prekrivanje mora delovati ne glede na pozicijo (odklon od trupa) rok in nog. Izboljšati želimo tudi delovanje v slabših svetlobnih pogojih.

Fizični svet okoli nas je tridimenzionalen. RGB podatkovni tok zaznava svet v dveh dimenzijah. Globinska kamera (globinski podatkovni tok) doda tretjo dimenzijo – globino. Oba podatkovna toka skupaj imenujemo barvno-globinski vizualni sistem [5]. Obe kameri moramo umeriti (kalibrirati) [6].

Naprave, ki so trenutno najbolj znane, imajo dober SDK vmesnik in so cenovno dostopne so:

- Structure senzor.
- Microsoft Kinect.
- Leap motion.

Le ena pa deluje na mobilni napravi: Structure senzor.

Poglavje 2

Globinska zaznava

2.1 Strojna oprema za zajem globinske slike

2.1.1 Kinect

Microsoft je na trg ponudil že dve generaciji Kinecta. Prvo generacijo je leta 2010 izdal za uporabo primarno na igralni konzoli Microsoft Xbox 360 [7]. Drugo nadgrajeno napravo je izdal leta 2014 skupaj z igralno konzolo Xbox One [8]. Napravi vidimo na sliki 2.1. Obe generaciji naprav ponujata priklop na osebni računalnik preko USB vtičnika.

Prva generacija za detekcijo globine uporablja tehnologijo strukturirane svetlobe. RGB podatkovni tok zajema z ločljivostjo 640×480 slikovnih elementov in s hitrostjo 30 slik na sekundo. Globinski podatkovni tok zajema z ločljivostjo 320×240 slikovnih elementov in s hitrostjo 30 slik na sekundo.

Območje globinskega zaznavanja je med 1,2 in 3,5 metri razdalje. Zaradi kompleksnosti prepoznave človeškega telesa omogoča sledenje največ dvema skeletonoma.

Druga generacija za detekcijo globine uporablja tehnologijo, ki deluje po načelu časa preleta svetlobnega pulza. Tehnologija omogoča uporabo tudi v odprtih prostorih. Druga generacija naprave barvni podatkovni tok zajema v HD ločljivosti 1920×1080 slikovnih elementov in s hitrostjo 30 slik na se-

(a)

(b)

Slika 2.1: Microsoft Kinect: (a) prva generacija, (b) druga generacija.

kundo. Globinski podatkovni tok zajema z ločljivostjo 512×424 slikovnih elementov in s hitrostjo 30 slik na sekundo.

Prva generacija ima čas procesiranja 90 ns, medtem ko ima druga generacija čas procesiranja 60 ns. Druga generacija podpira USB 3.0 tehnologijo.

Za zunanje razvijalce je Microsoft ponudil SDK [9]. Omogoča dostop do naprednejših funkcij – detekcija skeletona, detekcija kretenj, glasovno prepoznavanje, itd.

Kinect izvaja prepoznavanje skeletona v dveh korakih:

- Izgradnja sveta: Zajete podatke pretvori v virtualni 3D prostor.
- Strojno učenje: Izvajanje strojnega učenja z uporabo baze posnetkov skeletonov v različnih pozicijah.

Baza posnetkov človeških teles v različnih pozicijah je zelo obsežna. Posledično je rezultat detekcije zelo dober.

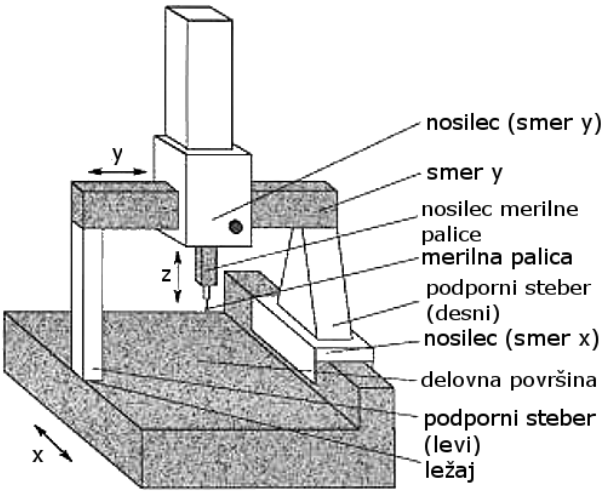
2.1.2 Structure senzor

Structure senzor je izdelek podjetja Occipital. V magistrski nalogi je Structure senzor predstavljal primarni vir globinskih podatkov. Podroben opis naprave, njenih značilnosti in SDK se nahaja v poglavju 3.

2.2 Tehnologije zajema

Poznamo 4 različne načine zajema globinske slike:

- Tipalo na robotski roki.
- Merjenje po načelu časa preleta svetlobnega pulza.
- Računalniško podprta tomografija.
- Optično merjenje.



Slika 2.2: Princip delovanja tipala na robotski roki [11].

Načini se med seboj razlikujejo po namembnosti, uporabljeni tehnologiji, ceni in natančnosti. Pri vseh načinih je potrebno umerjati napravo, glede na namen uporabe [10].

2.2.1 Tipalo na robotski roki

Tipalo na robotski roki (angl. coordinate measuring machine) deluje na osnovi fizičnega dotika. Robotska roka izvaja fizične dotike na objektu, ki ga želimo izmeriti/določiti. Na točki dotika odčitamo $[x, y, z]$ vrednosti koordinatnega sistema. S serijo meritev pod različnimi koti lahko v celoti zgradimo virtualni model objekta [10].

Princip delovanja senzorske roke vidimo iz opisa posameznih delov naprave (slika 2.2).

2.2.2 Merjenje po načelu časa preleta svetlobnega pulza

Metoda, ki meri po načelu časa preleta svetlobnega pulza meri čas med oddanim in prejetim signalom (angl. time of flight). Večina danes poznanih globinskih merilcev uporablja to tehnologijo. Komponente časovnega optičnega

bralnika so [12]:

- Osvetlitvena enota/laser: Za osvetlitev se uporablja infrardeča svetloba, ker je ta človeškemu očesu nevidna. Merjenje se izvaja z do 100 Mhz frekvence. Osvetlitvena enota uporablja LED ali laser diode, ker omogočajo doseganje želene frekvence.

- Optika: Prvotna naloga optike je filtriranje svetlobe z enako valovno dolžino kot jo oddaja osvetlitvena enota. S tem se minimizira šum na podatkih. Svetloba preko optičnega sistema potuje do svetlobnega senzorja.

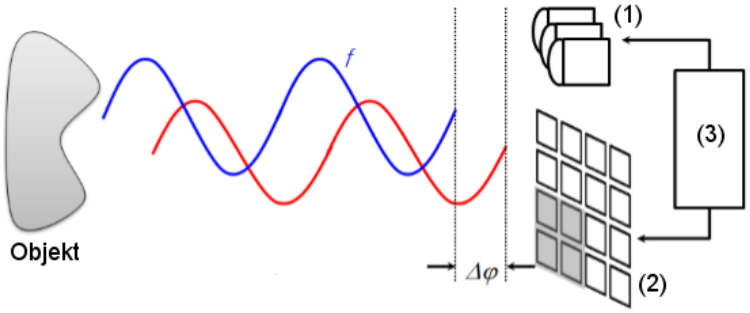
- Svetlobni senzor: Najpomembnejša enota, ki skrbi za merjenje časa svetlobnega signala. Svetlobni senzor je sinhroniziran z osvetlitveno enoto preko gonilnika.

- Gonilnik: Skrbi za povezavo med osvetlitveno enoto in svetlobnim senzorjem. Signali med eno in drugo enoto morajo potovati s čim manjšim časovnim zamikom. Časovni zamik ustvarja napako v izračunanih podatkih.

- Vmesnik: Skrbi za komunikacijo med zunanjo programsko kodo in napravo. Omogoča nastavljanje parametrov v napravi. Podatke transformira v strukturirano obliko in prenese izven naprave.

Merjenje razdalje se izvaja na podlagi pretečenega časa med oddanim in prejetim signalom [13]. Slika 2.3 prikazuje oddan signal (modra barva) s svetlobnega senzorja (oznaka 1) in prejeti signal (rdeča barva) na osvetlitveni enoti (oznaka 2). Medsebojno sta enoti povezani z gonilnikom (oznaka 3). Pretečen čas med oddanim in prejetim signalom (Δ_φ) je osnova za računanje razdalje [13].

Prednost metode je neobčutljivost na ambientalno svetlobo. To velja v primeru, da ambientalna svetloba nima enake frekvence kot jo oddaja osvetlitvena enota. Prednost je tudi fizična velikost. Naprave so manjše in



Slika 2.3: (1): Osvetlitvena enota, (2): svetlobni senzor, (3): gonilnik [13].

zaradi tega bolj okretne. Omogočajo merjenje večjih razdalj kot ostale metode (tipalo na robotski roki, računalniško podprta tomografija ali optična metoda).

Slabost metode je kompleksnost računanja. Za izvajanje potrebujemo zmogljivo procesorsko enoto. Kompleknost računanja prinaša manjšo ločljivost zajema podatkov kot pri ostalih metodah [14, 15].

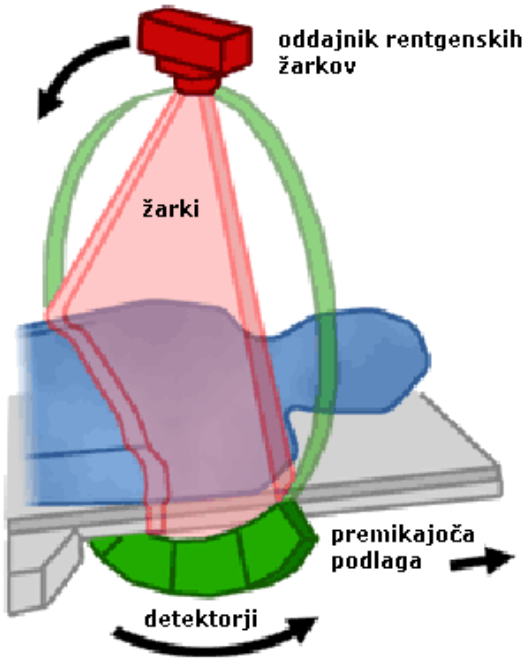
2.2.3 Računalniško podprta tomografija

Računališko podprto tomografijo največkrat srečamo v segmentu industrije in medicine. Prednost metode je pridobivanje informacij znotraj objekta in ne samo na površini. Tehnologija deluje na podlagi rentgenskih žarkov. Zaradi sevanja je tehnologija primerna za uporabo v kontroliranem okolju. Na sliki 2.4 vidimo uporabo tehnologije na področju medicine [10].

2.2.4 Optika

Optične metode merjenja globine delimo na dve podkategoriji:

- Aktivni optični merilci (projiciranje strukturirane svetlobe).
- Pasivne metode (stereo vid).



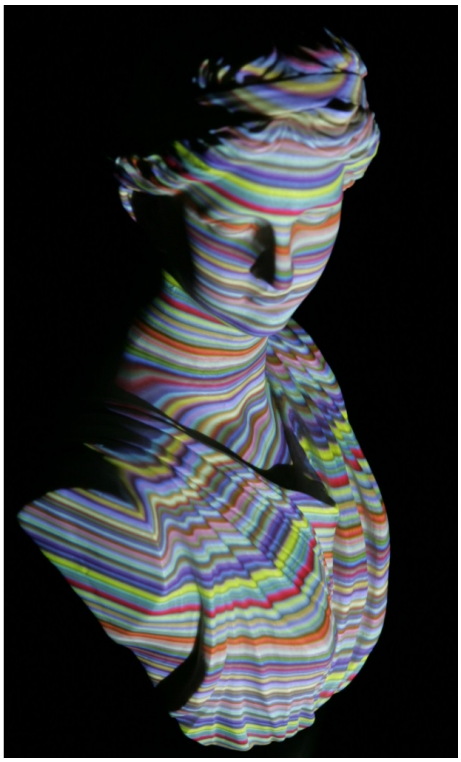
Slika 2.4: Uporaba računalniško podprte tomografije v medicini [16].

Metoda strukturirane svetlobe projicira svetlobo na površino v prostoru. Svetloba je projicirana v obliki vzorcev ali mrež. Pogosto uporabljamo projiciranje barvnih črt na površino. S tem pridobimo večjo natančnost merjenja. Običajno svetlobo projiciramo v svetlobnem spektru, ki je človeškemu očesu neviden.

Svetlobo projiciramo na površino z uporabo ene izmed dveh tehnologij:

- Projiciranje dveh laserjev, ki v časovnem obdobju generirata mrežo.
- Konstantno projiciranje vzorcev na površino.

Aktivni optični merilci so primerni za uporabo v zaprtih prostorih in na kratkih razdaljah. Ta metoda je občutljiva na močnejšo svetobo in ni primerna za detekcijo globine na površinah, ki odsevajo svetlobo. Njena prednost je nizka cena in enostavnost delovanja [17, 18]. Primer projekcije vidimo na sliki 2.5.



Slika 2.5: Projekcija barvnih črt na površino [19].

Poglavje 3

Structure senzor

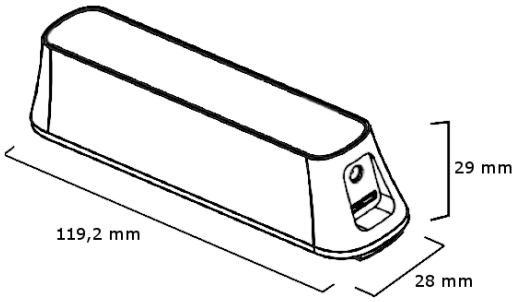
Structure senzor je izdelek podjetja Occipital [20], ustanovljenega leta 2013. Istega leta je podjetje pričelo s kampanjo zbiranja zagonskega kapitala na spletni strani za množično financiranje Kickstarter [21]. Kampanja je doživela velik uspeh in še istega leta so na trg poslali prve naprave.

Prvotni namen podjetja je bil razvoj naprave za merjenje globine in omogočiti uporabo na iOS operacijskemu sistemu. Kasneje je podjetje omogočilo uporabo na ostalih operacijskih sistemih z izdajo t.i. USB Hacker kabla.

Za upravljanje z napravo in procesiranje podatkov na iOS operacijskem sistemu so v podjetju razvili SDK. Structure SDK deluje izključno na iOS operacijskemu sistemu. Najbolj pomembe funkcije in podrobnosti SDK so predstavljene v poglavju 3.2.

Razvijalci so podprli integracijo z odprtokodno knjižnico OpenNI [22]. Knjižnica je namenjena procesiranju globinskih (3D) informacij. Po izidu Microsoft Kinect naprave je zanimanje za knjižnico OpenNI naraslo. Knjižnica OpenNI je postala vodilna na področju zajema 3D informacij.

Kasneje so razvijalci Structure senzorja podprli Unity [23] in SceneKit [24]. Unity je večplatformni igralni pogon. SceneKit je igralni pogon, napisan v Objective-C jeziku, in je namenjen poganjanju na OSX in iOS operacijskih sistemih.



Slika 3.1: Dimenzije Structure senzorja.

3.1 Karakteristike naprave

Structure senzor omogoča pritrditev z namenskimi nosilci na različne mobilne naprave. Podjetje Occipital ponuja nosilce za Apple iPad in Apple iPhone novejših generacij. Nosilce za ostale mobilne naprave dobimo na spletu. Zaradi namena uporabe na prenosnih napravah so dimenzije Structure senzorja temu primerne. Obliko in fizične dimenzije naprave vidimo na sliki 3.1.

Structure senzor izvaja branje okolice z ločljivostjo 640×480 ali 320×240 slikovnih elementov. Pri obeh ločljivostih omogoča hitrost skeniranja s 30 - imi slikami na sekundo (FPS). Pri manjši ločljivosti (320×240) SDK omogoča hitrost skeniranja s 60-imi slikami na sekundo. Vidno polje v horizontalni smeri je 58° in 45° v vertikalni smeri.

Naprava je optimizirana za merjenje objektov v oddaljenosti od 40 do 350 cm. V tabeli 3.1 so predstavljene pričakovane napake merjenja v odvisnosti od razdalje objekta.

3.2 Structure senzor SDK

Structure senzor SDK je spisan v jeziku Objective-C. Pobuda za razvoj samostojnega SDK je bila nezmožnost poganjanja OpenNI knjižnice na operacijskem sistemu iOS. SDK je razdeljen na dva dela:

razdalja [mm]	Relativna napaka [%]	Absolutna napaka [mm]
400	0,15	0,6
1000	0,3	3
2000	0,6	12
3000	1	30
3500	1,1	38,5

Tabela 3.1: Napaka merjenja v odvisnosti od razdalje.

- Nizkonivojski kontroler senzorja (angl. sensor controler), ki skrbi za dostop do toka informacij in njegovih nastavitev.

- Visokonivojski SLAM pogon (angl. simultaneous localization and mapping engine), ki omogoča sledenje in izvajanje 3D mapiranja.

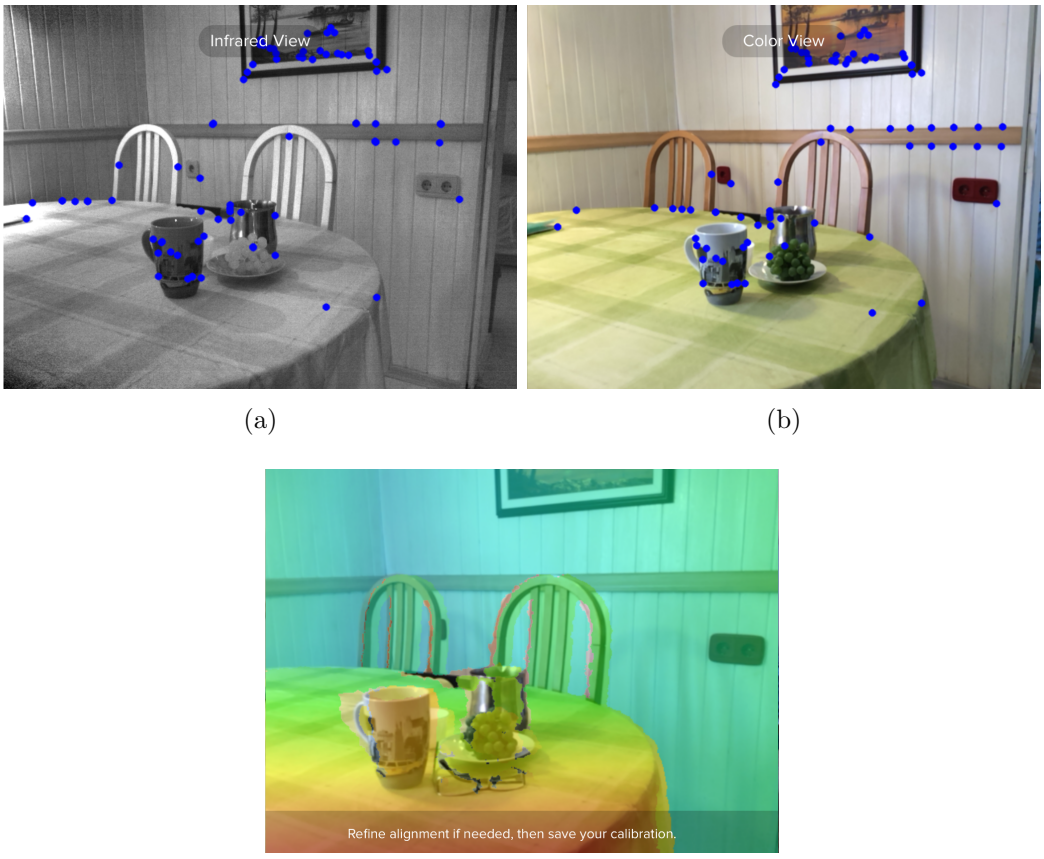
Structure senzor SDK omogoča upravljanje z RGB in globinskim podatkovnim tokom. SDK globinski podatkovni tok pridobiva z globinskega senzorja, RGB podatkovni tok pa z zadnje kamere iPada ali iPhona.

3.2.1 Umerjanje

Umerjanje (angl. calibration) je postopek, pri katerem zagotovimo ujemanje slikovnih elementov pri sinhronem branju RGB in globinskega podatkovnega toka. Samodejno ali ročno umerjanje se izvaja z detekcijo robov na obeh podatkovnih tokovih. Kvaliteto ujemanja merimo z relativnim ujemanjem robov. Bližje so si točke robov, boljša je kvaliteta ujemanja (slika 3.2).

Za umerjanje Structure senzorja je podjetje izdalo namensko aplikacijo. Na AppStore jo najdemo pod imenom Structure Sensor Calibrator (slika [25]). Rezultat umerjanja se shrani v Structure senzor, s katerim smo izvajali postopek umerjanja.

SDK hrani stanje umerjanja v spremenljivki *STCalibrationType*. Spremenljivka hrani več stanj:



(a) (b)

(c)

Slika 3.2: Zaslonski posnetki umerjanja: (a) globinski podatkovni tok, (b) RGB podatkovni tok, (c) ročne nastavitve ujemanja.

- STCalibrationTypeNone: S trenutnim Structure senzorjem umerjanje še ni bilo izvedeno.

- STCalibrationTypeApproximate: S trenutnim Structure senzorjem je bilo narejeno umerjanje s samodejnim algoritmom – lahko prihaja do večjih odstopanj med podatkovnima tokoma.

- STCalibrationTypeDeviceSpecific: S trenutnim Structure senzorjem je bila narejena kalibracija z aplikacijo Structure Sensor Calibrator. Umerjanje je izvedeno s samodejnim umerjanjem in za tem z ročnimi popravki [26].

3.2.2 Postavitev točke v 3D koordinatni sistem

Po končanem umerjanju lahko vsaki točki v globinskem podatkovnem toku določimo lokacijo v 3D prostoru. Z uporabo inercijskih senzorjev lahko določimo lokacijo vsake točke v prostoru, kljub premikanju kamere v prostoru.

Iz zapisa podatkov v globinskem podatkovnem toku $[i, j, z]$ naredimo pretvorbo v 3D koordinatni sistem $[x, y, z]$ z enačbami:

$$x = (i - \frac{w}{2}) * (z + minDistance) * scaleFactor \tag{3.1}$$

$$y = (j - \frac{h}{2}) * (z + minDistance) * scaleFactor \tag{3.2}$$

$$z = z, \tag{3.3}$$

kjer je:
$minDistance = -10$,
$scaleFactor = 0,0021$.

Enačbe in vrednosti ($minDistance$ in $scaleFactor$) predstavljajo najboljši

približek pri pretvorbi iz $[i, j, z]$ v $[x, y, z]$. Podatki za pretvorbo so vzeti iz dokumentacije za projekt OpenKinect [27].

3.2.3 Metode

V tem poglavju bomo predstavili nekaj najpomembnejših metod za vzpostavitev povezave in upravljanje s podatkovnimi tokovi [26]. Izsek programske kode razreda `Structure` vidimo v kodi 3.1:

Koda 3.1: Structure SDK metode.

```
1 - (STSensorControllerInitStatus)initializeSensorConnection;
2 - (NSString *)getName;
3 - (NSString *)getSerialNumber;
4 - (NSString *)getFirmwareRevision;
5 - (bool)startStreamingWithOptions:(NSDictionary *) error:(NSError *
      \_\_autoreleasing *);
6 - (void)sensorDidOutputDepthFrame:(STDepthFrame *);
```

- `initializeSensorConnection`: Funkcija skrbi za vzpostavitev povezave s Structure senzorjem. Metoda vrne `STSensorControllerInitStatusSuccess`, če je bila vzpostavitev uspešna. Če je povezava že vzpostavljena, metoda vrne `STSensorControllerInitStatusAlreadyInitialized`.
- `getName`: Vrne ime Structure senzorja.
- `getSerialNumber`: Vrne identifikacijsko številko Structure senzorja.
- `getFirmwareRevision`: Vrne verzijo programske opreme na Structure senzorju. V primeru, da verzija zaostaja za aktualno, jo posodobimo z novim SDK.
- `startStreamingWithOptions`: Funkcija začne postopek pridobivanja podatkov s podatkovnih tokov z nastavitvami, zapisanimi v strukturi `NSDictionary`. Izberemo podatkovne tokove, ki jih želimo pridobivati,

nastavimo ločljivost posameznih podatkovnih tokov, vključimo/izključimo sinhronizacijo podatkovnih tokov, vključimo/izključimo polnjenje lukenj zaradi anomalij v podatkih, nastavimo predvideno oddaljenost objektov (nastavitev fokusa) ipd.

- `sensorDidOutputDepthFrame`: Metoda povratnih klicev (angl. callback), ki vrača podatke iz globinskega toka podatkov. Podatki so organizirani v strukturo `STDepthFrame`.

- `ensorDidOutputSynchronizedDepthFrame`: Metoda povratnih klicev, ki vrača podatke iz globinskega in RGB toka podatkov. Podatki iz globinskega toka so organizirani v strukturo `STDepthFrame`, podatki iz RGB podatkovnega toka pa so organizirani v strukturo `STColorFrame`. Strukturi hranita podatke o širini, višini in izmerjenih globinah/barvah.

3.2.4 Anomalije v podatkih

Structure senzor je narejen za uporabo v notranjih prostorih. V okolju, kjer ozadje ni v dosegu globinskega senzorja, je delovanje nemogoče. Močenjša zunanja svetobla ali direktna sončna svetloba onemogoči napravi izvajanje meritev. Po narejeni analizi testnih primerov smo ugotovili, da manjši delež (okno v ozadju) slikovnih elementov, ki so izven dosega, ne vpliva na ostale slikovne elemente. V primeru, da je slikovnih elementov, ki so izven dosega več kot $\frac{1}{3}$ celotne slike, popači še ostale slikovne elemente. V tem primeru naprava postane neuporabna [28, 29].

Če prihaja do anomalij v podatkih samo v manjšem obsegu (posamezni slikovni elementi), jih lahko odpravimo z izvajanjem interpolacije v okolici.

Poglavje 4

Rekonstrukcija zaprtega prostora

Za namene spoznavanja s Structure senzorjem smo implementirali SLAM algoritem [30]. Aplikacija izvaja rekonstrukcijo zaprtega prostora in kasnejšo vizualizacijo le-tega.

4.1 RGB in RGB-D SLAM algoritem

RGB in RGB-D SLAM algoritma sta namenjena rekonstrukciji prostora in detekciji lokacije snemalne naprave v prostoru. RGB SLAM algoritem uporablja samo RGB podatkovni tok, medtem ko RGB-D SLAM uporablja še globinski podatkovni tok [31]. V praksi se izkaže, da algoritmi brez globinskega podatkovnega toka nimajo dovolj informacij. Posledično se tak algoritem v praksi obnese slabo [30].

4.2 Zajem in rekonstrukcija

V aplikaciji smo za rekonstrukcijo in ugotavljanje lokacije v prostoru uporabili visokonivojski objekt StructureSLAM. Objekt nam omogoča gradnjo prostora z RGB-D SLAM algoritmom. Za izgradnjo tekstur nam objekt

Slika 4.1: Primer gradnje zaprtega prostora z RGB-D SLAM algoritmom.

omogoča branje RGB podatkovnega toka. Za sledenje premikom po prostoru v treh smereh uporabljamo inercijske senzorje. Primer uporabe lahko vidimo na sliki 4.1.

Poleg zajema in rekonstrukcije smo implementirali shranjevanje rekonstruiranega prostora na spominski medij naprave. Rekonstruiran prostor shranimo v datoteko .obj s pripadajočimi teksturami in začetno orientacijo (odklon). Začetno orientacijo potrebujemo, ker želimo, da bo kasnejša vizualizacija imela enako orientacijo (primer: v kasnejši vizualizaciji rekonstruiranega prostora se severna stena nahaja na severni strani).

4.3 Koraki lokalizacije kamere v prostoru in rekonstrukcija prostora

SLAM algoritmi delujejo po principu primerjanja stanja s prejšnjimi iteracijami. Iskanje podobnosti med iteracijam je narejeno z algoritmom RANSAC. RANSAC algoritem uporabljamo na RGB ali globinskem podatkovnem toku.

Algoritem ima najboljše rezultate pri kombinaciji obeh podatkovnih tokov. RANSAC algoritem deluje po principu iskanja robov.

SLAM algoritmi iterativno izvajajo operacije v več korakih:

1. Inicializacija: Določitev prvotne lokacije v prostoru.
2. Predvidevanje: Na podlagi senzorskih meritev izračunamo novo predvideno lokacijo snemalne naprave.
3. Merjenje: Dodajanje novih podatkov za rekonstrukcijo prostora. Z RANSAC algoritmom na RGB in globinskem podatkovnem toku poiščemo robove. Primerjamo jih s sliko iz prejšnje iteracije. Novo lokacijo dobimo z družitvijo predvidene in izračunane razdalje.
4. Ponavljanje iteracij: Ponavljaj 2. in 3. korak, kolikor je potrebno [32].

4.4 Prikaz

V postopku vizualizacije rekonstruiranega prostora iz spominskega medija pridobimo .obj datoteko s pripadajočo teksturo. Poleg pridobimo še začetno orientacijo (odklon), izmerjen v predhodnem postopku snemanja. V virtualni prostor postavimo rekonstruiran prostor v realni velikosti. V center rekonstruiranega prostora postavimo kamero.

Za spreminjanje pogleda po vizualiziranem prostoru spreminjamo orientacijo iPada (uporabljamo inercijske senzorje). Za spreminjanje lokacije v prostoru uporabljamo smerne puščice. Pred prikazom je treba inercijske senzorje umeriti na pravilno smer neba. Inercijski senzor vedno začne z vrednostjo odklona (angl. yaw) 0. Ta odklon nato prilagodimo trenutni smeri neba. Za prilagoditev odklona potrebujemo dostop do kompasa. Primer sprehoda po vizualiziranem prostoru vidimo na sliki 4.2.

Slika 4.2: Sprehod po rekonstruiranem prostoru.

Poglavje 5

Detekcija človeka v realnem času

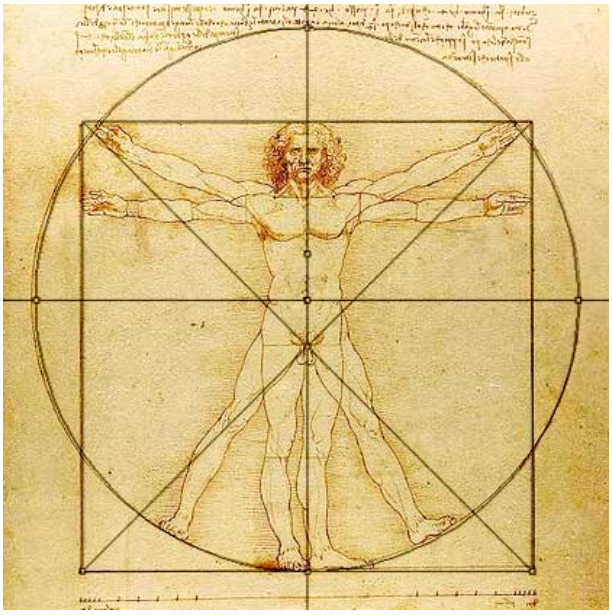
5.1 Fizikalne zakonitosti človeškega telesa

Deli človeškega telesa in njihova velikost so v medsebojno znanih razmerjih. S staranjem se ta razmerja spreminjajo. Eno izmed prvih študij razmerij delov človeškega telesa je naredil arhitekt Vitruvij. Razmerja človeškega telesa po Vitruviju je kasneje narisal Leonardo da Vinci (slika 5.2) [34].

Najbolj znana razmerja so:

- Razdalja med odročenima rokama je enaka višini človeka.
- Razdalja od vrha čela do dna brade je $\frac{1}{10}$ razdalje celotne višine.
- Razdalja od vrha glave do dna brade je $\frac{1}{8}$ razdalje celotne višine.
- Razdalja od kolen do gležnjev je $\frac{1}{4}$ razdalje celotne višine.
- Razdalja od komolcev do konca prstov na rokah je $\frac{1}{4}$ razdalje celotne višine [35].

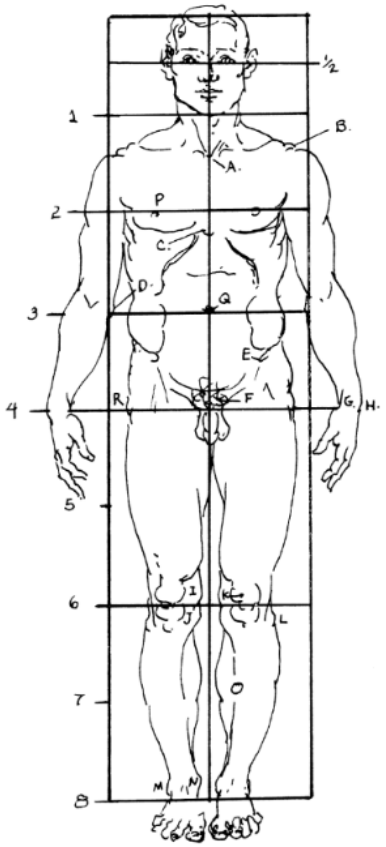
Poznamo tudi grško ali renesančno predstavitev rezmerij človeškega telesa [33]. To predstavitev človeškega telesa smo uporabili pri razvoju aplikacije.



Slika 5.1: Slika razmerij človeškega telesa po Vitruviju (avtor: Leonardo da Vinci).

Človeško telo po višini razdeli na 8 enakih delov. Razdelitev vidimo na sliki 5.2. Implementacija v praktičnem delu magistrske naloge se je opirala na naslednja razmerja:

- Glava se nahaja na najvišji poziciji in je visoka $\frac{1}{8}$ celotnega človeškega telesa.
- Ramena se nahajajjo na $\frac{1,5}{8}$ višine človeškega telesa.
- Roke so dolge $\frac{1}{2}$ višine človeškega telesa.
- Noge so dolge $\frac{1}{2}$ višine človeškega telesa.
- Kolki se nahajajo na poziciji $\frac{1}{2}$ višine človeškega telesa.



Slika 5.2: Renesančna predstavitev razmerij človeškega telesa [33].

5.2 Detekcija človeka z RGB podatkovnim tokom

Za tovrstno detekcijo potrebujemo RGB kamero. Prednost teh kamer je nizka cena, slabost pa je nezmožnost delovanja v slabših svetlobnih pogojih. Premalo svetlobe v prostoru povzroči nezmožnost zaznavanja barv na RGB senzorju. Preveč svetlobe (sončna svetloba, usmerjena svetloba v kamero) povzroči prenasičenost barv na senzorju.

5.2.1 Cartoob

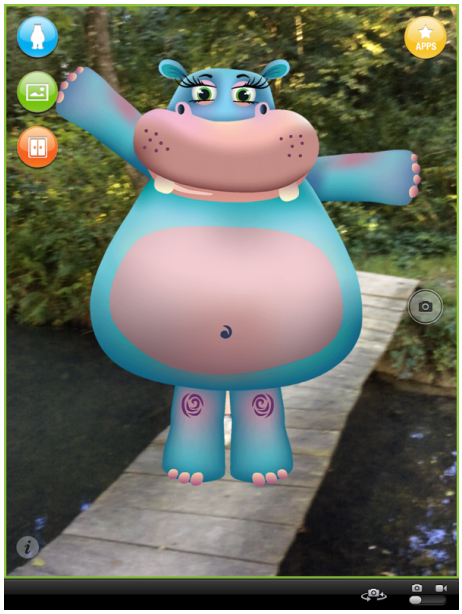
Cartoob [4] je serija aplikacij komercialne narave, dostopnih na Apple AppStore. Detekcija človeškega telesa je implementirana na podlagi RGB kamere.

Aplikacija na podlagi HOG podatkov naredi klasifikacijo. Iz učne množice poišče najbolj verjetno pozicijo glede na HOG značilke [36]. Učna množica vsebuje naslednje poze človeškega telesa:

- Roki: pozicija ob telesu, pravokotno na telo ali dvignjena.
- Noge: obe nogi na tleh, leva noga privzdignjena, desna noga privzdignjena.

Po enolično določeni pozi človeškega telesa algoritem nariše virtualni karakter. Velikost virtualnega karakterja je določena z detektirano velikostjo glave. Če je razlika med prejšnjo in trenutno sliko manjša od 5. slikovnih elementov, se položaj in velikost ne spremenita – s tem je odpravljen problem preskakovanja lika na sliki. Zaslonsko sliko aplikacije v delovanju vidimo na sliki 5.3.

Uspešnost algoritma je zelo odvisna od osvetlitve. V primeru slabe osvetlitve nam aplikacija javi opozorilo.



Slika 5.3: Aplikacija Cartoob po uspešni zaznavi človeškega telesa [4].

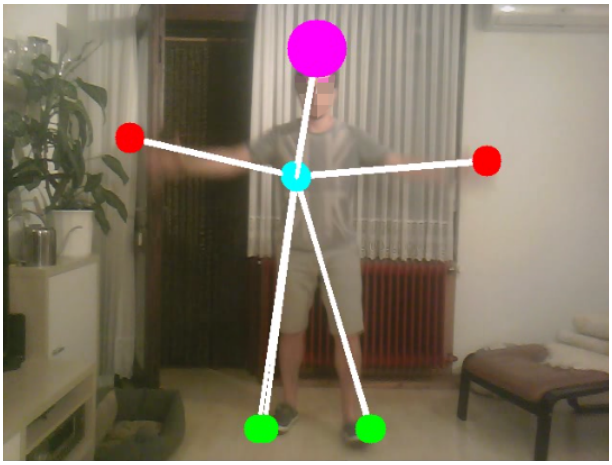
5.2.2 Algoritem iskanja premikajočih objektov

Algoritem iskanja premikajočih objektov [37] uporablja RGB kamero. Algoritem je pogosto implementiran v gonilnikih cenovno ugodnih spletnih kamer ali v kamerah na prenosnih računalnikih. Algoritem za svoje delovanje potebuje kamero na statični poziciji.

Algoritem premikajočih objektov in iskanja osebe se izvaja v štirih korakih:

1. Branje RGB podatkovnega toka in zapis podatkov v seznam zadnjih prebranih slik (seznam vsebuje slike zadnjih štirih iteracij).

2. Sledi postopek primerjanja zadnje prebrane slike s seznama slik prejšnjih iteracij. Na pozicijah slikovnih elementov, pri katerih se ugotovi neujemanje (z upoštevanjem tolerance zaradi netočnosti kamer) se naredi nova slika zaznamkov. S slike zaznamkov izbrišemo slikovne elemente, ki bi glede na parameter bili premajhni za človeško telo.



Slika 5.4: Algoritem iskanja premikajočih objektov s kamero na prenosnem računalniku.

3. Na sliki zaznamkov poiščemo robne točke (z algoritmom za odkrivanje robov). Nato poiščemo potencialne točke rok, nog in glave. Potencialne točke predvidimo glede na pozicijo na podlagi razmerij človeškega telesa.

4. S seznama robnih in potencialnih točk izberemo tiste, ki se ujemajo z razmerji človeškega telesa. Manjkajoče točke nadomestimo s tistimi iz prejšnih iteracij.

Prednosti algoritma so hitro delovanje, enostavna implementacija in velika razširjenost. Slabost algoritma je nedelovanje v slabih svetlobnih pogojih in hitrih spremembah svetlobe. Algoritem zaradi iskanja sprememb v iteracijah ni primeren za uporabo na mobilnih napravah – premikanje naprave. Testiranje algoritma na cenovno ugodni kameri (vgrajeni v prenosni računalnik) vidimo na sliki 5.4.

5.3 Detekcija človeka z globinskim podatkovnim tokom

Za tovrstno detekcijo potrebujemo globinsko kamero. Detekcija človeškega telesa z globinsko kamero je praviloma bolj uspešna od detekcije z RGB kamero. Globinska kamera ni odvisna od svetlobnih pogojev, barve ozadja ali barve oblačil.

5.3.1 Prepoznava delov človeškega telesa z uporabo baze posnetkov poz človeškega telesa

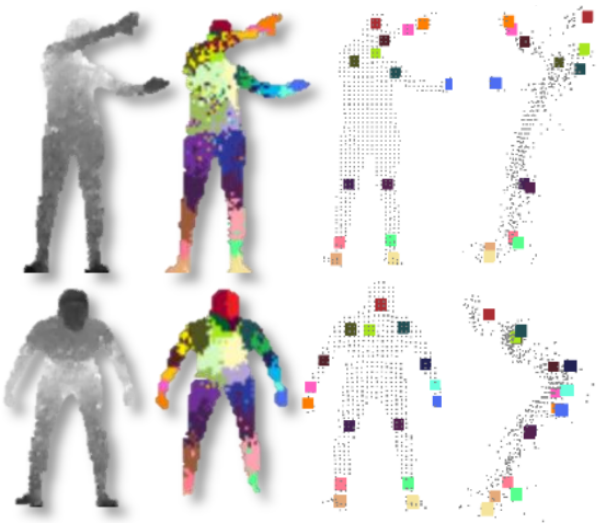
Prepoznava delov človeškega telesa z uporabo baze posnetkov poz človeškega telesa je metoda, ki jo uporablja Microsoft Kinect za prepoznavo človeškega telesa [38]. Metoda temelji na veliki podatkovni bazi vnaprej posnetih poz človeškega telesa. Poze so posnete v različnih rotacijah, velikostih, pričeskah, spolu, starosti ipd. [39, 40]. Postopek prepoznave:

1. Ločevanje telesa od ozadja. Osnovna detekcija telesa je narejena z detekcijo glave. Sosednje točke dodajamo k telesu z računanjem razdalj.

2. Prvotno označevanje telesa na podlagi 31-ih delov telesa: glava (4. deli), vrat, ramena (2 dela), roka (4. deli), zapestje (2 dela), komolec (2 dela), trup (4. deli), noga (4. deli), koleno (2 dela), gleženj (2 dela), stopalo (4. deli).

3. Na podlagi označitev telesa izvajanje klasifikacije z uporabo naključnega gozda (angl. randomized decision forest). Ta drevesa so logična izbira zaradi velike baze vnaprej posnetih poz človeškega telesa. Vsaka iteracija prične iskanje z izbiro naključnega korena podatkovne baze. Naključni odločitveni gozdovi so primerni za izvajanje na grafičnem procesorju.

4. Končna enolična določitev poze. Poza ima definirane pozicije vseh 31-ih delov telesa v 3D prostoru [40].



Slika 5.5: Klasifikacija človeškega telesa s pomočjo Kinecta, kjer je vsak del človeškega telesa označen s svojo barvo [40].

Rezultat algoritma prepoznave delov človeškega telesa s pomočjo Kinecta vidimo na sliki 5.5.

5.3.2 Detekcija s poznavanjem fizikalnih zakonitosti človeškega telesa

Algoritem deluje brez podatkovne baze vnaprej posnetih poz človeškega telesa. Prednost neuporabe baze je manjša poraba prostora na spominskem mediju. Algoritem so avtorji zasnovali in predstavili v članku [41]. Algoritem smo implementirali v 6-ih korakih:

1. Gradnja objektov v 3D prostoru: Na podlagi globinske slike algoritem kreira objekte v prostoru. Izbiranje pripadnosti posameznim objektom določimo na podlagi maksimalne dovoljene razdalje ($dis_{\max}$). Vsako točko pridružimo objektu, ki se nahaja na krajši razdalji od trenutne točke. Če so vse razdalje večje od $dis_{\max}$, kreiramo nov objekt.

2. Izbor objekta, ki predstavlja človeško telo: Na podlagi zaznanega objekta

človeškega telesa iz prejšnje iteracije vzamemo središčno točko (točka v trupu telesa) z oznako T. Objekt, ki je najbližji točki T, postane potencialni kandidat. Če v prejšnji iteraciji nismo našli objekta človeškega telesa, je točka T postavljena na sredino slike. Točko T postavimo na sredino slike tudi pred prvim iskanjem človeškega telesa – prejšnja iteracija ne obstaja.

3. Določitev ekstremnih točk: Na podlagi detekcije iz prejšnje iteracije vzamemo točko, ki je definirala vrh glave. Tej točki poiščemo najboljši približek na trenutni iteraciji. Če definirana točka vrha glave ne obstaja, prvi ekstrem določimo z določitvijo točke, ki je najbolj oddaljena od točke T. Od te točke poiščemo najdaljšo razdaljo za določitev druge ekstremne točke. Tretjo ekstremno točko dobimo z iskanjem točke, ki je najdlje oddaljena od prve in druge ekstremne točke.

4. Iskanje točke, ki definira vrh glave: Ekstremne točke preverjamo po vrsti in iščemo točko, ki potencialno definira vrh glave. S preverjanjem pozicije na objektu ugotavljamo ali ekstremna točka definira vrh glave. Če smo v prejšnji sliki našli objekt človeškega telesa, je vedno prvi ekstrem tudi točka, ki definira vrh glave.

5. Iskanje ostalih definiranih točk: Na podlagi širine glave in pozicije na objektu lahko najdemo še ostale točke:

   - *Ramena*: Širino ramen pridobimo z računanjem razmerij človeškega telesa in podatka o širini glave. Višino ramen pridobimo s pregledom objekta.
   - *Dlani*: Dlani poiščemo na najbolj oddaljeni levi in desni strani objekta ter s sledenjem objekta v smeri od ramen navzven.
   - *Komolci*: Komolce dobimo na podlagi razmerij človeškega telesa z merjenjem razdalje med rameni in dlanmi.
   - *Kolki*: Pozicijo kolkov dobimo na podlagi razmerij človeškega telesa.

- *Stopala*: Stopala poiščemo na najnižjih točkah objekta, ki si sledijo od kolkov navzdol.
- *Kolena*: Kolena dobimo na podlagi razmerij človeškega telesa z merjenjem razdalje med kolki in stopali.

6. Shranjevanje atributov: Zadnji korak je nov izračun točke T. Točko T in točko, ki definira vrh glave shranimo v pomnilnik za naslednjo iteracijo.

Poleg implementacije algoritma iz članka, smo mi implementirali odpravo preskakovanja lika na sliki z upoštevanjem točk, ki smo jih našli v prejšnjih iteracijah. Algoritem je optimiziran in omogoča večnitno izvajanje. Odprtokodna implementacija tega algoritma za Microsoft Kinect na operacijskem sistemu Microsoft Windows je dostopna na naslovu [42].

5.4 Detekcija človeka s sinhronim RGB in globinskim podatkovnim tokom

Detekcija človeka s sinhronim RGB in globinskim podatkovnim tokom prinese prednosti obeh podatkovnih tokov. Slabost uporabe obeh podatkovnih tokov je večja časovna kompleksnost zaradi večjega nabora podatkov.

Prva možnost implementacije je detekcija človeškega telesa na obeh tokovih posebej in združevanje rezultatov na koncu. Drugi način je prepletanje detekcije človeškega telesa na obeh tokovih skozi celoten algoritem. V tem primeru združitev na koncu ni potrebna. Algoritem prepletanja obeh podatkovnih tokov smo uporabili tudi v naši aplikaciji. Opis algoritma je predstavljen v poglavju 6.

Poglavje 6

Razvoj aplikacije

6.1 Strojna in programska oprema

Aplikacija je testirana za Apple iPad Mini 2 [43]. RGB podatkovni tok pridobivamo s kamere, nameščene na zadnji strani naprave. Globinski podatkovni tok pridobivamo s Structure senzorja [44]. Dostop do podatkov je narejen s Structure Sensor SDK [26].

Logika aplikacije je spisana v C++ jeziku [45]. Izbira C++ jezika nam omogoča enostavno prehodnost med platformami. Ogrodje programske kode je napisano v Objective-C jeziku [46] ki, je zelo razširjen programski jezik v operacijskih sistemih OSX in iOS.

Za izvajanje manipulacij s podatkovnimi tokovi uporabljamo programsko knjižnico OpenCV [47].

6.2 Algoritem iskanja človeškega telesa

6.2.1 Detekcija obraza

Na RGB podatkovnem toku in z uporabo HAAR kaskadnega klasifikatorja poskušamo pridobiti lokacijo in velikost obraza [48]. Uspešno iskanje obraza je obvezen predpogoj za nadaljnje izvajanje algoritma. Učinkovito iskanje

obraza na celotni RGB sliki se je izkazalo za potratno operacijo, ki ni primerna za izvajanje v realnem času. Pri manjši natančnosti iskanja in slabši ambientalni svetlobi algoritem ni našel koordinat obraza. Naredili smo optimizacijo iskanja obraza v dveh korakih:

1. Iskanje z večjo natančnostjo smo naredili na enakih koordinatah in v 2× večjem območju od območja obraza v prejšnji sliki. Če v tem območju ne najdemo obraza, ga pričnemo iskati na območju celotne slike z zmanjšano natančnostjo (slika 6.1a).

2. Iskanje obraza z veliko natančnostjo pogosto povzroči veliko lažno pozitivnih primerov, ki so običajno zelo majhne velikosti (slika 6.1b). Vsako potencialno območje, ki je znotraj drugega potencialnega območja, se izbriše. Zatem algoritem s seznama potencialnih kandidatov izbere tistega, ki ima največjo površino. Ostale kandidate izbriše (slika 6.1c).

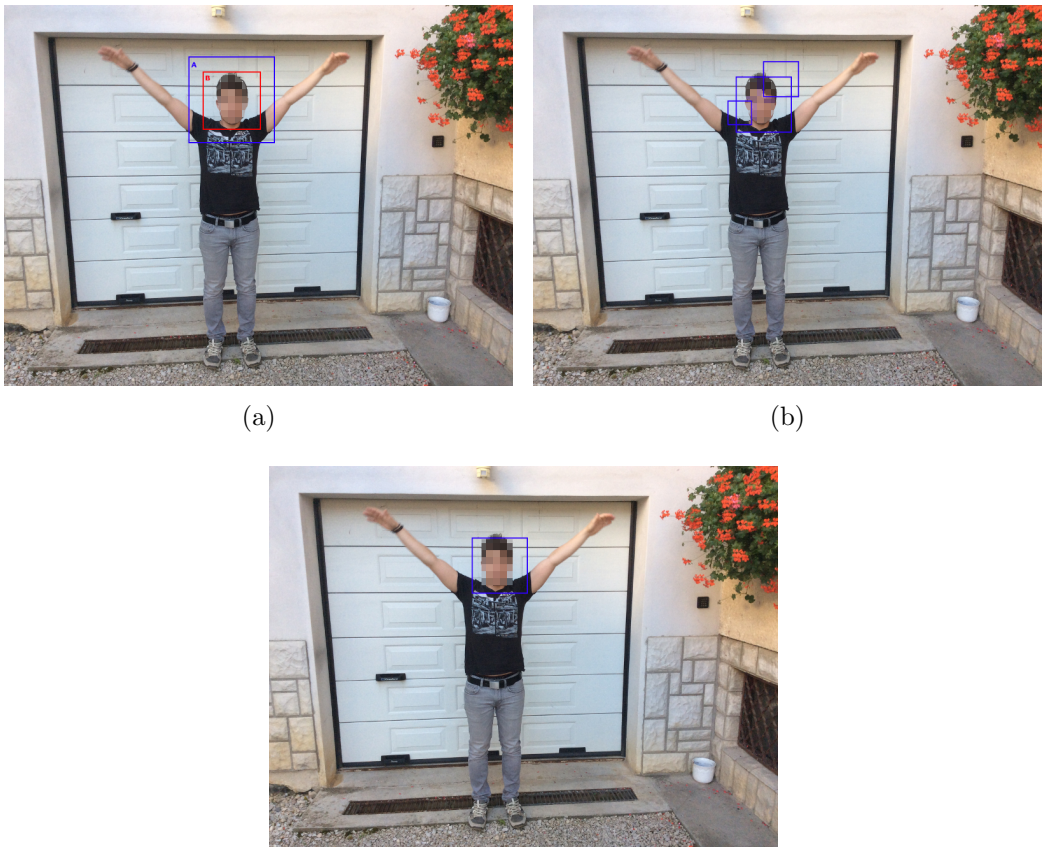
6.2.2 Določitev območja zgornjih okončin

Določitev območja zgornjih okončin se izvaja ob prehodu z RGB podatkovnega toka na globinski tok podatkov. Kot izhodišče vzamemo najden obraz iz prejšnjega koraka. Območje povečamo po vnaprej znanih skalarjih v vseh smereh 2-dimenzionalne slike. Skalarji so pomnožene vrednosti območja obraza.

Skalarji in njihova usmeritev na 2-dimenzionalni sliki:

- V smeri navzgor za skalar 2,5.
- V smeri na levo za skalar 3,5.
- V smeri na desno za skalar 3,5.
- V smeri navzdol za skalar 3.

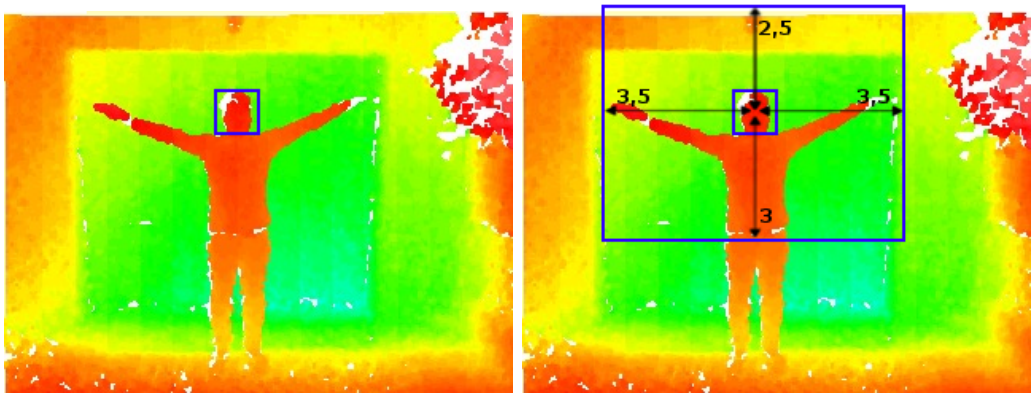
Vrednosti skalarjev smo določili na podlagi renesančne predstavitve razmerij telesa (poglavje 5.1). Rezultat razširjenega območja si lahko ogledamo na sliki 6.2.



(a) (b)

(c)

Slika 6.1: Detekcija obraza: (a) območje iskanja z večjo natančnostjo, (b) najdena potencialna območja obraza in (c) izbira najboljšega kandidata.



(a) (b)

Slika 6.2: Grafična predstavitev območja rok.

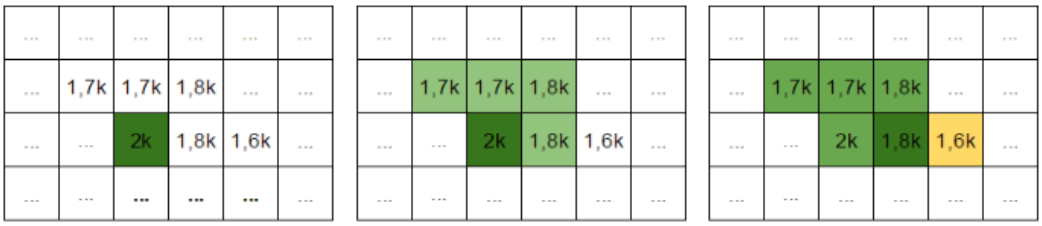
Za določitev spodnje meje območja smo uporabili skalar 3. S tem skalarjem povečamo območje do višine kolkov. Dolžina rok sega nižje od višine kolkov. Ta problem rešujemo naknadno, z računanjem kota med pozicijo ramen in zadnjo znano pozicijo rok na razširjenem območju.

RGB in globinski podatkovni tok imata različni ločljivosti vhodnih podatkov. Razširjeno območje zapišemo glede na celotno RGB sliko in ga prenesemo na globinski podatkovni tok.

6.2.3 Binarizacija območja zgornjih okončin in izvajanje podatkovne redukcije

Binarizacijo območja zgornjih okončin izvajamo na globinskem podatkovnem toku. Binarizacijo smo implementirali z algoritmom poplavljanja (angl. flood fill) [49]. Algoritem poplavljanja loči točke med seboj glede na kriterij pripadnosti. Če točka zadostuje kriteriju pripadnosti, jo označimo s pozitivno vrednostjo, v nasprotnem primeru z negativno vrednostjo:

- Prvi del kriterija pripadnosti se izvaja z računanjem razlike globin med točkami in njenimi sosedi. Damo ji oznako Z_{disMax}.
- Drugi del kriterija pripadnosti je maksimalna dovoljena razlika med



(a) (b) (c)

Slika 6.3: Algoritem poplavljanja s parametroma $Z_{\text{disMax}} = 200$ in $Z_{\text{disMaxInit}} = 300$: (a) izsek globinske slike, (b) preverjanje referenčne točke in njenih sosedov (svetlo-zelene točke), (c) preverjanje druge točke in njenih sosedov – točka z vrednostjo 1,6k (rumena točka) ne izpolnjuje drugega kriterija (razlika med referenčno in rumeno točko je večja od 300).

referenčno in trenutno točko. Označimo jo z $Z_{\text{disMaxInit}}$. Če velja $Z_{\text{disMax}} \leq Z_{\text{disMaxInit}}$, točko označimo s pozitivno vrednostjo.

Na sliki 6.3 vidimo primer poplavljanja z upoštevanjem kriterija pripadnosti.

Referenčna točka je osnovni podatek o zaznani razdalji človeškega obraza. Algoritem poplavljanja začne izvajati algoritem v referenčni točki.

V tem koraku izvajamo redukcijo podatkov. Redukcijo izvajamo s spuščanjem stolpcev in vrstic za vrednost redukcije. Vsaka točka dobi nove sosede, ki so oddaljeni za korak redukcije. Pozicije sosedov točke, ki se nahaja na lokaciji $[x, y]$ z redukcijo red so:

- Levi sosed: $[x - red, y]$.
- Desni sosed: $[x + red, y]$.
- Zgornji sosed: $[x, y - red]$.
- Spodnji sosed: $[x, y + red]$.

6.2.4 Postopek tanjšanja

Nad binarizirano sliko zgornjih okončin izvajamo morfološko operacijo tanjšanja. Tanjšanje je iterativna operacija, pri kateri debelino zaznanega območja

(a) (b)

Slika 6.4: Tanjšanje: (a) binarska slika, (b) slika po končanem algoritmu tanjšanja.

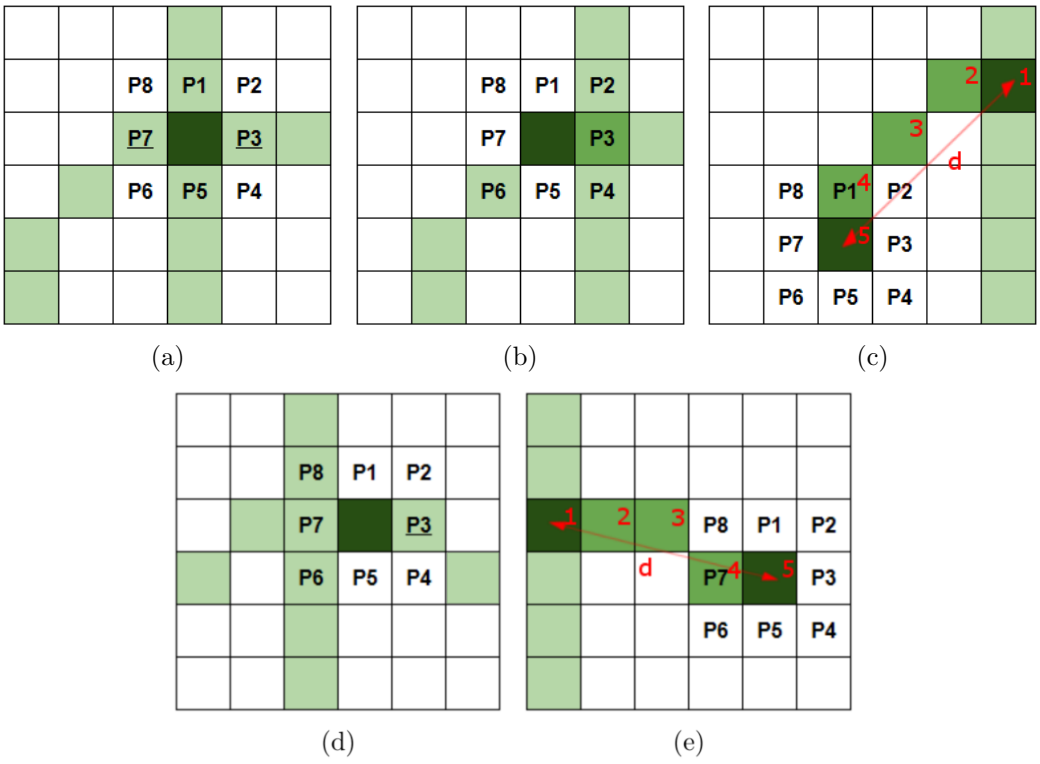
P8	P1	P2
P7		P3
P6	P5	P4

Slika 6.5: Označevanje sosednjih točk.

zmanjšamo na debelino ene točke. S postopkom tanjšanja si v naslednjih korakih pridobimo bolj učinkovito prepoznavanje pozicije trupa in zgornjih okončin. Tanjšanje izvajamo z Zhang-Suen algoritmom za tanjšanje [50]. Uporabili smo odprtokodno C++ implementacijo, dostopno na GitHub-u [51]. Primer tanšanja vidimo na sliki 6.4.

6.2.5 Detekcija zgornjih okončin

V sliki stanjšanih zgornjih okončin in trupa poiščemo najvišjo točko skeleta na sliki. Točko iščemo v območju glave iz prvega koraka. S tem pripomoremo k časovni optimizaciji algoritma. Za nadaljevanje izvajanja označimo sosede vsake nerobne točke slike z oznakami P1 do P8 (slika 6.5). Od najvišje do najnižje točke iterativno sledimo črti. V vsaki iteraciji naredimo 4 korake, ki



Slika 6.6: Postopek iskanja rok.

so predstavljeni tudi v grafični obliki na sliki 6.6:

1. Preštejemo sosede s pozitivno vrednostjo na točkah od P1 do P8. Če je število sosedov 3 ali več, smo našli stičišče in nadaljujemo z izvajanjem preostalih korakov. V nasprotnem primeru preostale korake preskočimo in se prestavimo na novo točko (slika 6.6a).

2. Za detekcijo leve roke preverimo nahajanje pozitivne vrednosti na pozicijah od P6 do P8. To točko imenujemo potencialna točka leve roke. Za detekcijo desne roke preverimo nahajanje pozitivne vrednosti na pozicijah od P2 do P4. To točko imenujemo potencialna točka desne roke.

3. Želimo se prepričati, da sta potencialni točki leve ali desne roke stičišči trupa in rok. Preprečiti želimo lažno pozitivne primere, pri katerih je potencialna točka samo preskok črte po algoritmu tanjšanja. Prestavimo se na potencialno točko in pri potencialni točki leve roke preverimo vrednosti na pozicijah P1 in od P5 do P8. Pri potencialni točki desne roke preverimo vrednosti na pozicijah od P1 do P5. Vsaj ena točka mora biti pozitivna in potencialna točka roke postane koren (ramena) roke (sliki 6.6b in 6.6d).

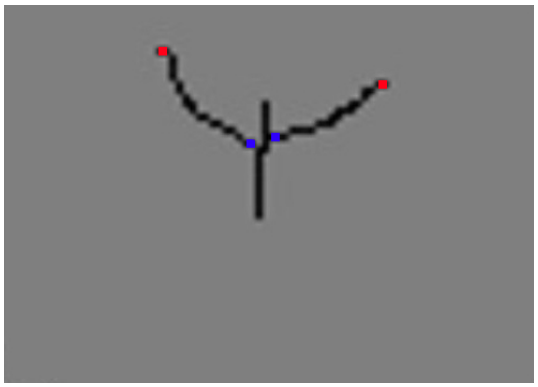
4. Iterativno sledimo črti roke. Ko pridemo do konca, izračunamo razdaljo med stičiščem in končno točko. Če je razdalja med točkama od 3-kratnika do 5-kratnika velikosti glave, točko shranimo in zapišemo kot levo/desno zapestje (sliki 6.6c in 6.6e).

Rezultat detekcije zgornjih okončin vidimo na sliki 6.7.

6.2.6 Detekcija spodnjih okončin

Detekcija kolkov in nog se izvaja v dveh korakih:

1. Iskanje rok je zaključeno na višini kolkov. S tem podatkom pridobimo informacijo o višini kolkov. Za iskanje pozicije levega kolka na binarni



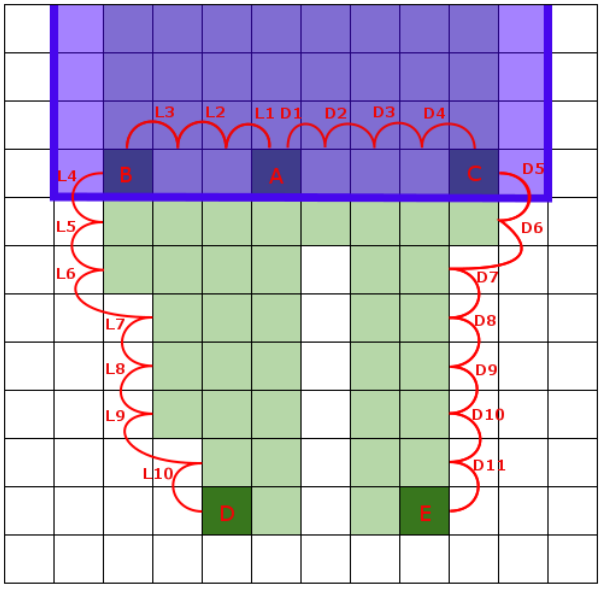
Slika 6.7: Stanjšana slika z najdenimi rameni (modra barva) in zapestji (rdeča barva).

sliki rekurzivno izvajamo premike v levo smer, dokler je izpolnjen pogoj pozitivne vrednosti trenutne točke. Pozicijo desnega kolka dobimo na enak način, s premikom v desno smer. Obe točki shranimo kot levi/desni kolk. Točki se nahajata na prehodu med pozitivno in negativno vrednostjo v binarni sliki.

2. Izhodišče za iskanje stopal sta najdeni točki kolkov. Iz izhodiščne točke se iterativno premikamo po y koordinati navzdol in iščemo preskok med pozitivno in negativno vrednostjo. Iščemo v območju P4 do P6. Točka za novo iteracijo je tista, ki si lasti pozitivno vrednost, kot to prikazuje slika 6.8. Postopek ponavljamo do iteracije, ko ne najdemo več preskoka. Renesančna predstavitev razmerij človeškega telesa definira dolžino nog kot 4-kratnik velikosti glave. Preverimo ali je dolžina med kolki in stopali med 3,5 in 4,5-kratnikom. Če se nahaja v teh okvirjih, shranimo točko kot levo/desno stopalo.

6.3 Slikovni atlas

Slikovni atlas (angl. sprite sheet) je združitev več manjših ločenih slik v eno sliko. Slikovni atlasi so najbolj razširjena oblika hranjenja tekstur v 2D



Slika 6.8: Detekcija lokacije kolkov, nog in stopal.

grafiki. Koncept ni odvisen od izbire operacijskega sistema ali programskega jezika. Vsebina slikovnega atlasa je opisana v ločeni datoteki (običajno .XML datoteka) [52, 53]. Uporaba slikovnega atlasa prinaša naslednje prednosti [54]:

- Hitrost procesiranja: Sliko samo enkrat (na začetku) naložimo v grafični pomnilnik. Med delovanjem grafični kartici ni potrebno preklapljati med teksturami.
- Priprava slikovnega atlasa: Zaradi razširjenosti uporabe slikovnega atlasa je na trgu veliko programov, ki nam omogočajo hitro in kvalitetno izdelavo.
- Uporabnost: Enostavna izdelava več grafičnih podob za enako uporabnost (primer: več preoblek za en animirani lik).
- Učinkovite spremembe: Enostavno upravljanje in spreminjanje tekstur. Spremembe brez posegov v programski kodi.

Slika 6.9: Slikovni atlas, ki je bil uporabljen v naši aplikaciji.

Slabosti uporabe slikovnega atlasa [54]:

- Prostor v pomnilniku: Zaradi velikosti je nalaganje slik s teksturami v pomnilnik prostorsko potratno.
- Ni primerno za uporabniško generirano vsebino: Če uporabnik sam sestavlja podobo animiranega lika in ima veliko grafičnih možnosti, uporaba slikovnega atlasa ni najbolj primerna izbira.

Primer slikovnega atlasa, ki smo ga uporabili v naši aplikaciji, lahko vidimo na sliki 6.9. Glavni metodi za definicijo slikovnega atlasa sta zapisani v kodi 6.1:

Koda 6.1: Metodi za delo s slikovnim atlasom.

```
1    void setOrigTextureSprite(cv::Mat matrix);
2    void setTexturePart(int order, SkeletonTextureParts part, int x,
         int y, int defaultWidth, int defaultHeight, double
         defaultRotation, std::vector<JointPositionObjects *>
         jointsPosition)
```

- `setOrigTextureSprite`: Metoda kot argument sprejema OpenCV matriko (3 kanali) s sliko, ki predstavlja slikovni atlas.

- `setTexturePart`: S spremenljivko `order` nastavimo vrstni red izrisa. S spremenljivko `part` nastavimo, kateri del telesa predstavlja trenutna tekstura. S spremenljivkami `x`, `y`, `defaultWidth` in `defaultHeight` nastavimo lokacijo in velikost teksture. `DefaultRotation` nastavimo, ko želimo, da ima tekstura prvotno rotacijo. V seznam `jointsPosition` nastavimo imena in pozicije sklepov na posamezni teksturi.

6.4 Algoritem prekrivanja

Po končanem postopku iskanja človeškega telesa dobimo seznam devetih točk. V primeru, da algoritem ne najde katerega izmed parov rama-dlan ali kolk-stopalo, je seznam lahko krajši. V primeru, da algoritem ne najde osnovnih razmerij, vrne prazen seznam. Iskane točke človeškega telesa so:

- Vrh glave.
- Leva rama.
- Leva dlan.
- Desna rama.
- Desna dlan.
- Levi kolk.
- Levo stopalo.
- Desni kolk.
- Desno stopalo.

Na podlagi teh točk smo naredili prekrivanje z animiranim likom. Algoritem prekrivanja deluje v naslednjih korakih:

1. Pridobivanje informacij o teksturi: Vrstni red izrisa posamezih delov telesa je pomemben zaradi prekrivanja tekstur na sklepih (ramena, kolki). Tekstura, ki jo izrišemo, bo na njeni celotni površini prekrila kar je izrisano pod njo. V naši knjižnici vrstni red določimo ob inicializaciji posamezne teksture (funkcija `setTexturePart` – atribut `order`). V tem koraku izberemo po vrsti najmanjši še neizrisan del človeškega telesa.

2. Pridobivanje teksture: Izbran del človeškega telesa iz prve točke in podatkov o koordinatah, širini in višini dela, pridobimo iz slikovnega atlasa. Sliki z uporabo barvne maske odstranimo ozadje (naredimo transparentno).

3. Izvajanje transformacij: Na podlagi izračunanih podatkov izvedemo spreminjanje velikosti in rotacije. Rotacije tekstur izvajamo okoli vnaprej znanih točk, ki smo jih določili ob definiciji posamezne teksture.

   Na teksturi rok izvajamo rotacijo na ramenski točki. Teksturo nog rotiramo na točki kolka. Trup in glava imata rotacijsko točko v središču teksture. Velikost trupa določimo z merjenjem razdalje med rameni in kolki. Pozicijo trupa določimo z najboljšim približkom izmerjenih ramen in ramen v teksturi.

   Velikost teksture rok, nog in glave določimo glede na velikost trupa. Rotacijo rok dobimo z izračunom kota med rameni in dlanmi. Rotacijo nog dobimo z izračunom kota med kolki in stopali.

4. Izris: Na sliko, ki smo jo dobili z RGB podatkovnega toka izrišemo teksturo na ustrezno lokacijo z upoštevanjem transformacij iz koraka 3.

Poglavje 7

Rezultati

Kvalitativno in kvantitativno analizo smo naredili na podlagi štirih scenarijev. S scenariji smo testirali vse aspekte, ki so pomembni pri detekciji človeškega telesa. Scenariji se delijo na podscenarije. Na vsakem podscenariju smo preverili delovanje algoritmov na 9 pozah človeškega telesa.

Med seboj smo primerjali metodo z RGB podatkovnim tokom, metodo z globinskim tokom in metodo, ki združuje oba podatkovna toka.

Pri metodi, ki uporablja globinski podatkovni tok in metodi, ki združuje oba podatkovna toka, smo povprečni odzivni čas merili z namenskimi funkcijami v programski kodi. Pri metodi z RGB podatkovnim tokom smo odzivni čas merili s snemanjem zaslona in kasnejšo analizo v programih za video obdelavo. To metodo smo uporabili, ker nismo imeli dostopa do programske kode. Brez dostopa ne moremo uporabiti namenskih funkcij za merjenje povprečnega odzivnega časa.

7.1 Poze človeškega telesa

Vsak scenarij (ali podscenarij) smo testirali na 9 pozah človeškega telesa (slika 7.1). Nekaj poz smo izbrali naključno, nekaj pa smo jih izbrali na podlagi klasifikacije Cartoob algoritma. Poze iz Cartoob smo izbrali za lažjo primerjavo s to metodo.

Slika 7.1: Poze človeškega telesa na katerih smo merili rezultate posameznega algoritma.

(a) (b)

Slika 7.2: Scenarij urbanega okolja: (a) temnejša oblačila, (b) svetlejša oblačila.

7.2 Scenariji

Štirje testirani scenariji so bili:

- Urbano okolje: Lokacija scenarija je v urbanem okolju pred garažnimi vrati. Za preverjanje kvalitete procesiranja smo z različnimi oblačili izvedli dva podscenarija. Ozadje je bilo v dosegu globinskega senzorja (slika 7.2).

- Mestni park: Lokacija scenarija je bila v mestnem parku ob obzidju. Ozadje je bilo zelo podobno barvi kože in v dosegu globinskega senzorja (slika 7.3).

- Naravno okolje: Lokacija je bila v naravi ob jezeru. Na lesenem mostu smo izvedli dva podscenarija z različnimi oblačili. Scenarij je preverjal delovanje algoritma, kjer je ozadje dlje od dosega globinskega senzorja (slika 7.4).

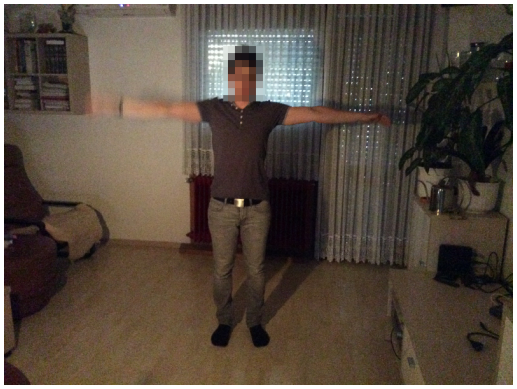
- Notranji prostor: Lokacija scenarija je bila notranji prostor. Pri tem scenariju z umetno svetlobo smo preverjali učinkovitost detekcije ob

Slika 7.3: Scenarij mestnega parka.

(a) (b)

Slika 7.4: Scenarij naravnega okolja: (a) temnejša oblačila, (b) svetlejša oblačila.

(a)



(b)

(c)

Slika 7.5: Scenarij notranjega prostora: (a) difuzna umetna svetloba, (b) delno zatemnjen prostor, (c) zatemnjen prostor.

različnih svetlobnih pogojih. Ozadnje je bilo v dosegu globinskega senzorja (slika 7.5).

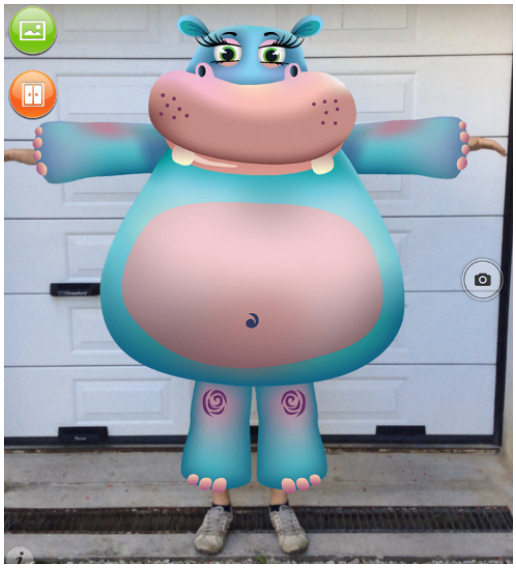
7.2.1 Scenarij 1: Urbano okolje

Analiza je pokazala, da v scenariju urbanega okolja barva oblačil nima vpliva (tabela 7.1).

Cartoob je imel ob začetku snemanja težave s premajhno količino svetlobe na senzorju. Kasneje se ta problem ni več pojavil. Cartoob je pravilno detektiral pozicijo v 66 % primerov. Detekcija nikoli ni bila pravilna ob

(a)

(b)

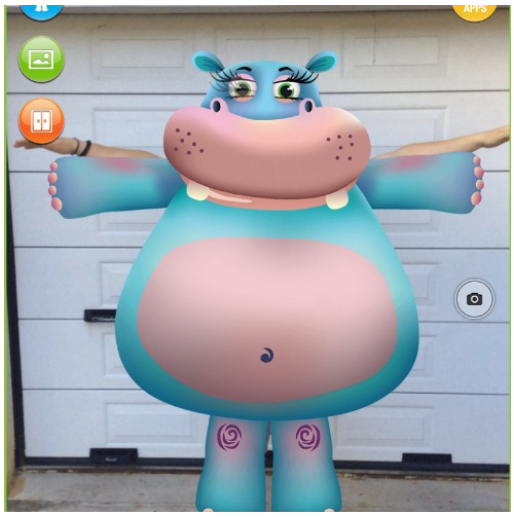


(c)

Slika 7.6: Scenarij urbanega okolja v svetlejših oblačilih: (a) algoritem z obema podatkovnima tokoma, (b) algoritem z globinskim podatkovnim tokom, (c) Cartoob.

(a)

(b)



(c)

Slika 7.7: Scenarij urbanega okolja v temnejših oblačilih: (a) algoritem z obema podatkovnima tokoma, (b) algoritem z globinskim podatkovnim tokom, (c) Cartoob.

obeh privzdignjenih rokah (pozicija 1 na sliki 7.1). Prav tako detekcija ni bila uspešna, ko je bila ena roka privzdignjena, druga pa spuščena ob telo (poziciji 3 in 5 na sliki 7.1). Povprečni odzivni čas je bil konstanten 41 ms (24 FPS).

Algoritem, ki uporablja samo globinski podatkovni tok, je pravilno detektiral pozo v manj kot polovici primerov. Posledično algoritem ni izrisal teksture ali pa je bila tekstura izrisana na napačni poziciji z napačno velikostjo. Povprečni odzivni čas je bil med 850 in 900 ms in je prevelik za uporabo v realnem času.

Algoritem z obema podatkovnima tokoma je učinkovito detektiral človeško telo. Algoritem ni prepoznal samo ene pozicije (pozicija 9 na sliki 7.1). Odzivni čas je okoli 100 ms in je še primeren za uporabo v realnem času (sliki 7.6 in 7.7).

Cartoob izvaja prekrivanje na osnovi klasifikacije iz množice 27 predefiniranih poz telesa. Naš algoritem izvaja prekrivanje na osnovi posameznih delov telesa in deluje neodvisno od kota med trupom in rokami oziroma nogami. Kvaliteta prekrivanja in posledično uporabniška izkušnja je s tem veliko boljša.

podscenarij/algoritem [pravilno detektirane klasifikacije (čas izvajanja)]	Cartoob (RGB p.t.)	Globinski p.t.	RGB & globinski p.t.
1. temnejša oblačila	66 % (41ms)	44 % (880ms)	88 % (110ms)
2. svetlejša oblačila	66 % (41ms)	44 % (850ms)	88 % (100ms)

Tabela 7.1: Rezultati scenarija v urbanem okolju.

7.2.2 Scenarij 2: Mestni park

Scenarij mestnega parka ima poudarek na detekciji človeškega obraza v okolju, kjer je barva ozadja podobna barvi kože. Pri tem scenariju nismo uporabili obeh tipov oblačil, ker oblačila ne vplivajo na postopek iskanja obraza. Scenarij se je obnašal podobno kot scenarij v urbanem okolju. Lisasto ozadje

podscenarij/algoritem [pravilno detektirane klasifikacije (čas izvajanja)]	Cartoob (RGB p.t.)	Globinski p.t.	RGB & globinski p.t.
1. temnejša oblačila	66 % (41ms)	22 % (950ms)	88 % (130ms)

Tabela 7.2: Rezultati scenarija v mestnem parku.

kamnitega zidu in podobna barva, kot je barva kože, ni izrazito poslabšala detekcije človeškega telesa (tabela 7.2).

Cartoob je pravilno detektiral pozicijo v 66 % primerov. Detekcija nikoli ni bila pravilna ob obeh privzdignjenih rokah (pozicija 1 na sliki 7.1). Prav tako detekcija ni bila uspešna, ko je bila ena roka privzdignjena druga pa spuščena ob telo (poziciji 3 in 5 na sliki 7.1). Povprečni odzivni čas je bil konstanten 41 ms (24 FPS).

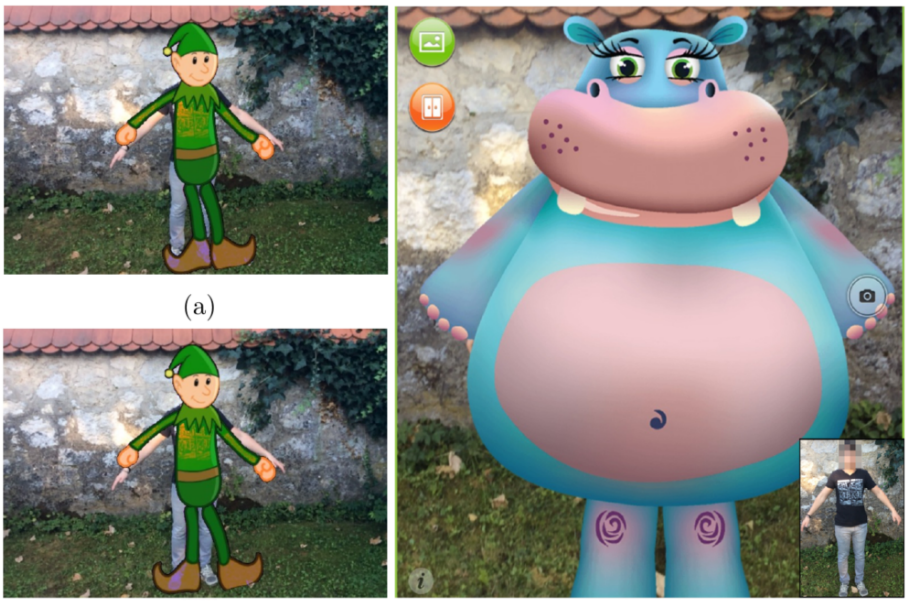
Algoritem, ki uporablja samo globinski podatkovni tok je pravilno detektiral 2 pozi. V večini primerov je telo našel na ozadju. Posledično je bila tekstura izrisana z napačno velikostjo in napačnimi koti med trupom in rokami/nogami. Povprečni odzivni čas je bil 950 ms in je prevelik za uporabo v realnem času.

Algoritem z obema podatkovnima tokoma je učinkovito detektiral človeško telo. Algoritem ni uspešno prepoznal samo ene poze (pozicija 9 na sliki 7.1). Odzivni čas je okoli 130 ms in je še primeren za uporabo v realnem času. Odzivni čas je nekoliko slabši kot pri prejšnjem scenariju, ker je algoritem iskanja obraza našel več potencialnih kandidatov (slika 7.8).

7.2.3 Scenarij 3: Naravno okolje

Scenarij naravnega okolja se je pri Cartoob izkazal enako učinkovito kot pri prvem in drugem scenariju. Algoritmi, pri katerih uporabljamo globinski senzor, so se izkazali slabo (tabela 7.3).

Cartoob je pravilno detektiral pozicijo v 7 primerih. Detekcija nikoli ni bila uspešna ob obeh privzdignjenih rokah (pozicija 1 na sliki 7.1). Prav tako detekcija ni bila uspešna, ko je bila ena roka privzdignjena, druga pa spuščena



Slika 7.8: Scenarij v mestnem parku: (a) algoritem z obema podatkovnima tokoma, (b) algoritem z globinskim podatkovnim tokom, (c) Cartoob.

ob telo (pozicija 5 na sliki 7.1). Povprečni odzivni čas je bil konstanten, 41 ms (24 FPS).

Algoritem, ki uporablja samo globinski podatkovni tok, se je izkazal zelo slabo. Detekcija ni bila pravilna v nobenem primeru. Povprečni odzivni čas je bil več kot 3000 ms. Zaznani objekt (kjer naj bi bila pozicija telesa) je bil zelo velik in posledično je bil čas procesiranja visok.

Algoritem z obema podatkovnima tokoma prav tako ni detektiral človeškega telesa. Povprečni odzivni čas je bil 35 ms. Čas je bil boljši kot pri ostalih scenarijih, ker je algoritem kmalu prenehal z detekcijo posameznih delov človeškega telesa.

Branje globinskega podatkovnega toka v tem scenariju ni bilo uspešno. Močnejša zunanja svetloba ali direktna sončna svetloba Structure senzorju onemogoči izvajanje meritev. Problem smo bolj natančno predstavili v poglavju 3.2.4. Naš algoritem je pravilno detektiral obraz (z RGB podatkovnega toka). Ko se je algoritem prestavil v korak izvajanja na globinskem podatkovnem toku, se je detekcija ustavila zaradi anomalij v podatkih. Primer globinskega podatkovnega toka vidimo na sliki 7.9.

podscenarij/algoritem [pravilno detektirane klasifikacije (čas izvajanja)]	Cartoob (RGB p.t.)	Globinski p.t.	RGB & globinski p.t.
1. temnejša oblačila	77 % (41ms)	0 % (3000ms)	0 % (35ms)
2. svetlejša oblačila	77 % (41ms)	0 % (3800ms)	0 % (40ms)

Tabela 7.3: Rezultati scenarija v naravnem okolju.

7.2.4 Scenarij 4: Notranji prostor

Scenarij notranjega prostora ima poudarek na delovanju v slabših svetlobnih pogojih. Pri tem scenariju nismo uporabili obeh tipov oblačil, ker je pogoj za delovanje (algoritma, ki uporabljata RGB podatkovni tok) uspešno najden obraz. Barva oblačil ne vpliva na postopek iskanja obraza.

Slika 7.9: Anomalije na globinskem podatkovnem toku Structure senzorja (črna barva predstavlja nedefinirane vrednosti). Desno spodaj je podana vhodna slika.

podscenarij/algoritem [pravilno detektirane klasifikacije (čas izvajanja)]	Cartoob (RGB p.t.)	Globinski p.t.	RGB & globinski p.t.
1. difuzna umetna svetloba	66 % (41ms)	44 % (3000ms)	88 % (35ms)
2. delno zatemnjen prostor	22 % (41ms)	33 % (3800ms)	88 % (40ms)
3. zatemnjen prostor	0 % (41ms)	33 % (3800ms)	33 % (40ms)

Tabela 7.4: Rezultati scenarija v notranjem prostoru.

Scenarij notranjega prostora nazorno pokaže razliko med RGB in globinsko zaznavo v slabših svetlobnih pogojih (tabela 7.4).

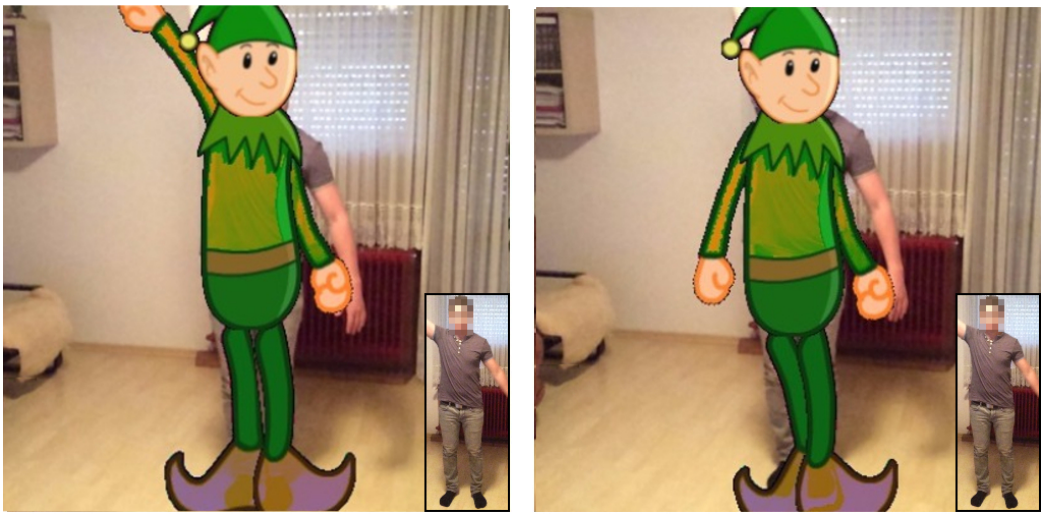
Cartoob je pri dnevni svetlobi detektiral 6 poz, pri delno zatemnjenem prostoru 2, pri zatemnjenem prostoru pa nobene. V vseh primerih nas je aplikacija opozarjala in svetovala, naj povečamo količino svetlobe. Pri delno osvetljenem prostoru je aplikacija našla obraz, a je bila pozicija telesa v polovici primerih napačna. Povprečni odzivni čas je bil konstanten, 41 ms (24 FPS).

Algoritem, ki uporablja samo globinski podatkovni tok se je izkazal podobno kot pri prvem in drugem scenariju. Časi so bili prav tako podobni, med 850 in 900 ms.

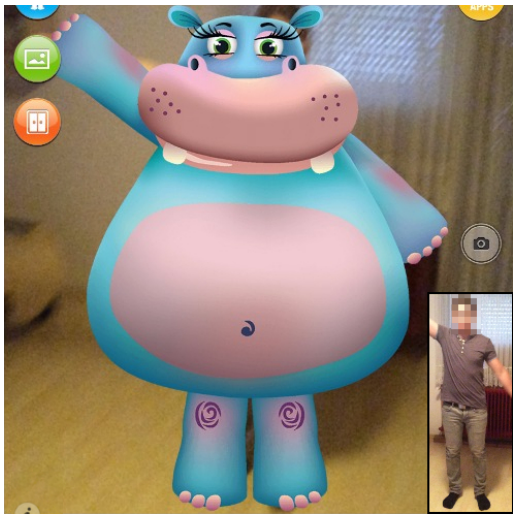
Algoritem z obema podatkovnima tokoma je učinkovito detektiral človeško telo pri difuzni umetni svetlobi in delno zatemnjenem prostoru. V teh dveh podscenarijih algoritem ni prepoznal samo 2. poze (3 in 9 na sliki 7.1). Pri detekciji človeškega telesa v zatemnejenem prostou je detektiral samo 3 poze. Povprečni odzivni časi so se gibali okoli 100 ms. Pri slabi osvetlitvi so bili povprečni odzivni časi slabši za 20 %.

Algoritem z obema podatkovnima tokoma se je pri vseh treh podscenarijih obnesel bolje od Cartoob algoritma.

Pri difuzni umetni osvetlitvi so se vsi algoritmi izkazali dobro (slika 7.10), pri srednje močni osvetlitvi se je naš algoritem izkazal dobro. Cartoob in algoritem z globinskim podatkovnim tokom sta bila brez uspeha (slika 7.11). Pri popolnoma zatemnjenem prostoru iskanje ni bilo uspešno z nobenim algoritmom.



(a) (b)



(c)

Slika 7.10: Scenarij zaprtega prostora z difuzno umetno svetlobo:(a) algoritem z obema podatkovnima tokoma, (b) algoritem z globinskim podatkovnim tokom, (c) Cartoob.

Slika 7.11: Scenarij zaprtega prostora s srednje dobro osvetlitvijo: algoritem z obema podatkovnima tokoma.

7.3 Skupni rezultat

Skupni rezultat je celotni pregled vseh scenarijev z vsemi algoritmi. Pregled je pokazal, da naš algoritem bolje detektira človeško telo v različnih okoljih in z več pozami kot Cartoob. Naš algoritem je manj občutljiv na svetlobo kot algoritem v Cartoob. Združevanje obeh podatkovnih tokov je dobra izbira.

Cartoob izvaja prekrivanje na osnovi predefinirane množice 27. poz. Naš algoritem izvaja prekrivanje na osnovi posameznih delov človeškega telesa. Število poz ni omejeno. Detektiran odklon udov od telesa se v enakem razmerju prenese na izris tekstur delov človeškega telesa. Zaradi tega je natančnost prileganja veliko boljša kot pri uporabi Cartoob. S tem je tudi uporabniška izkušnja bolj pristna.

Slabost našega algoritma so slabši povprečni odzivni časi. Cartoob izvaja eno iteracijo detekcije človeškega telesa in animacijo konstantno z 41 ms. Naš algoritem potrebuje okoli 100 ms.

Naš algoritem ima z uporabo Structure senzorja slabe rezultate v okoljih,

kjer ozadje sega izven maksimalnega dosega globinskega senzorja. Če je namreč razdalja večja od maksimalnega dosega naprave, se na sliki pojavljajo nedefinirana območja, ki segajo tudi v območja, v katerih bi naprava morala zaznati globino. Do teh anomalij prihaja, če je nedefinirano območje večje od $\frac{1}{3}$ velikosti celotne slike. To je ključna slabost Structure senzorja.

Do anomalij v podatkih prihaja tudi, kjer nimamo nedefiniranih območij, vendar imajo posamezni slikovni elementi nedefinirane vrednosti. Te anomalije niso kritične in jih lahko minimiziramo z izvajanjem interpolacije v okolici.

V našem algoritmu smo zaznali nekoliko večje trepetanje izrisanega lika kot v Cartoob algoritmu. Trepetanje smo v večji meri odpravili z računanjem razlike med novo in prejšnjo sliko. Teksturo smo izrisali po novih podatkih, ko smo zaznali spremembno večjo od 5. enot (slikovnih elementov ali stopinj v rotaciji).

Poglavje 8

Zaključek

V nalogi smo implementirali obogateno resničnost gibanja uporabnika v realnem času. S pomočjo podatkov RGB in globinskega podatkovnega toka smo implementirali detekcijo človeškega telesa – okostja. Detekcijo smo izvajali v realnem času s pomočjo fizikalnih zakonitosti človeškega telesa. Na podlagi detekcije smo prikazali animirani lik (z uporabo slikovnega atlasa), ki je kar najbolje prekril detektirano človeško telo.

Za klasifikacijo nismo uporabljali baze vnaprej posnetih slik človeškega telesa. S tem smo omogočili boljše prekrivanje brez omejitev, ki jih prinaša baza.

Structure senzor SDK ne vsebuje metod, ki bi omogočale detekcijo človeškega telesa. Uporaba knjižnic OpenNI in Kinect SDK je onemogočena na iOS operacijskem sistemu, zato je naša knjižnica je napisana v jeziku C++ za čim večjo prehodnost med platformami – deluje tudi na iOS operacijskem sistemu.

Naredili smo kvalitativno in kvantitativno analizo implementiranega algoritma ter ga primerjali z algoritmom, ki je vključen v aplikacijo Cartoob in algoritmom, ki uporablja samo globinski podatkovni tok.

Naš algoritem se je izkazal bolje od Cartoob algoritma. Prekrivanje izvaja na osnovi posameznih delov telesa (ne samo 27 poz, kot to dela Cartoob). Uporabniška izkušnja in kvaliteta prekrivanja je s tem veliko boljša. Zaradi

izvajanja na dveh podatkovih tokovih so bili časi procesiranja nekoliko slabši, kot pri Cartoob. Pri slabših svetlobnih pogojih se je naš algoritem izkazal bolje, ker je pri detekciji uporabljal tudi globinski podatkovni tok. Na zunanji sončni svetlobi Structure senzor ne deluje. Posledično naš algoritem ne najde človeškega telesa. Dokazali smo, da globinski senzorji v mobilnih napravah prinašajo nov nivo uporabniške izkušnje.

8.1 Nadaljnje delo

Naša pozornost je bila namenjena detekciji človeškega telesa s sprednje strani. Z globinskim senzorjem bi lahko naredili detekcijo tudi z drugih strani.

V prihodnosti bi lahko detektirali več sklepov (tudi komolce, kolena ipd.).

Dodatna optimizacija bi pripomogla k hitrejšemu delovanju algoritma in posledično boljši uporabniški izkušnji.

Aplikacija bi za objavo v trgovini aplikacij potrebovala še dodatne grafične in zvočne elemente, ki jih vsebuje aplikacija Cartoob.

Literatura

[1] (2016) Laboratorij za računalniški vid. Dostopno na: https://www.fri.uni-lj.si/si/laboratoriji/lrv/9857/project.html.

[2] A. Verri, E. Trucco, Introductory techniques for 3-D computer vision, str. 178–194, New Yersey, Prentice Hall, 1998.

[3] (2016) Samovozeče vozilo. Dostopno na: https://www.google.com/selfdrivingcar/how/.

[4] (2016) Cartoob spletna stran. Dostopno na: http://cartoobapp.com/.

[5] L. Shao, J. Han, P. Kohli, Z. Zhang, Computer vision and machine learning with RGB-D sensors, Basel: Springer, 2014.

[6] Urad RS za meroslovje, Mednarodni slovar izrazov zakonskega meroslovja, 2016.

[7] (2016) Microsoft Kinect za Xbox 360. Dostopno na: http://www.xbox.com/en-us/xbox-360/accessories/kinect.

[8] (2016) Microsoft Kinect za Xbox One. Dostopno na: http://www.xbox.com/en-us/xbox-one/accessories/kinect.

[9] (2016) Microsoft Kinect SDK. Dostopno na: https://developer.microsoft.com/en-us/windows/kinect.

[10] P. Peer, Iz dveh v tri razsežnosti, Moj mikro, let. 16, št. 2, str. 78–79, 2000.

[11] C. Dong, C. Zhang, B. Wang, G. Zhang, Reducing the dynamic errors of coordinate measuring machines, 125 (4), str. 831–839, 2010.

[12] B. Hagebeuker, A 3D time of flight camera for object detection, PMD Technologies GmbH, Siegen, 2007.

[13] M. Hansard, S. Lee, O. Choi, R. P. Horaud, Time-of-flight cameras: principles, methods and applications, Berlin: Springer Science, 2012.

[14] A. Kadambi, R. R. Ayush Bhandari, 3D depth cameras in vision: Benefits and limitations of the hardware, Cham: Springer Science, 2014.

[15] H. Sarbolandi, D. Lefloch, A. Kolb, Kinect range sensing: Structured-light versus Time-of-Flight Kinect, Computer Vision and Image Understanding, 139 (1), str. 1–20, 2015.

[16] (2016) Simulacija rentgenskih žarkov. Dostopno na: https://www.rpi.edu/dept/radsafe/public_html/gpu_project/index_gpu.html.

[17] K. Kutulakos, E. Steger, A theory of refractive and specular 3D shape by light-path triangulation, International Journal of Computer Vision, 76 (1), str. 13–29, 2008.

[18] J. Wilm, O. V. Olesen, R. Larsen, Slstudio: Open-source framework for real-time structured light, International Conference on Image Processing Theory, Tools and Applications (IPTA), str. 830–844, 2014.

[19] S. Narasimhan, S. J. Koppal, S. Yamazaki, Temporal dithering of illumination for fast active vision, European Conference on Computer Vision, str. 830–844, 2008.

[20] (2016) Leap motion senzor. Dostopno na: https://www.leapmotion.com.

[21] (2016) Spletna stran za množično financiranje Kickstarter. Dostopno na: https://www.kickstarter.com.

[22] (2016) OpenNI knjižnica. Dostopno na: http://openni.ru.

[23] (2016) Večplatformni grafični pogon Unity. Dostopno na: https://unity3d.com.

[24] (2016) SceneKit API in knjižnica za izrisovanje 3D grafike na iOS. Dostopno na: https://developer.apple.com/library/ios/documentation/scenekit/reference/scenekit_framework.

[25] (2016) Aplikacija Structure sensor calibrator v trgovini na AppStore. Dostopno na: https://itunes.apple.com/us/app/structure-sensor-calibrator/id914275485?mt=8.

[26] (2016) Structure senzor SDK dokumentacija. Dostopno na: https://developer.structure.io/sdk.

[27] (2016) Open Kinect dokumentacija, lokacija v 3D prostoru. Dostopno na: https://openkinect.org/wiki/imaging_information.

[28] (2016) Structure SDK forum: Anomalije v podatkih pri uporabi naprave v odprtih prostorih. Dostopno na: http://forums.structure.io/t/scanning-outside/3592/18.

[29] (2016) Structure SDK forum: Anomalije v podatkih pri uporabi naprave na neposredni sončni svetlobi. Dostopno na: http://forums.structure.io/t/scanning-outdoors-in-daylight-or-direct-sunlight/1645/8.

[30] N. Brunetto, S. Salti, N. Fioraio, T. Cavallari, L. Stefano, Fusion of inertial and visual measurements for RGB-D SLAM on mobile devices, International Conference on Computer Vision Workshop (ICCVW), str. 1–9, Washington, 2015.

[31] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments, International Journal of Robotics Research (IJRR), 31 (51), str. 647–663, 2012.

[32] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard, An evaluation of the RGB-D SLAM system, International Conference on Robotics and Automation (ICRA), str. 1691–1696, St. Paul, 2012.

[33] J. Sheppard, Anatomy: A complete guide for artists, New York: Watson–Guptil publicitaions, 1992.

[34] (2016) Slika Vitruvian Man (Gallerie dell' accademia). Dostopno na: http://www.gallerieaccademia.org/the-museum/?lang=en.

[35] (2016) Definicija slike Vitruvian Man. Dostopno na: http://leonardodavinci.stanford.edu/submissions/clabaugh/history/leonardo.html.

[36] R. Kreslin, Cartoob aplikacija, tehnična dokumentacija, 2013.

[37] (2016) Dokumentacija projekta Commodity tracking (Skeleton Tracking). Dostopno na: http://bobbybee.github.io/commoditytracking/.

[38] (2016) Sledenje uporabniku z metodo Skeletal Tracking. Dostopno na: https://msdn.microsoft.com/en-us/library/jj131025.aspx.

[39] L. Xia, C.-C. Chen, J. K. Aggarwal, Human detection using depth information by kinect, Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference, str. 15–22, Rapid City, 2011.

[40] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in

parts from single depth images, Communications of the ACM, 56 (1), str. 116–124, 2013.

[41] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, C. Theobalt, A data-driven approach for real-time full body pose reconstruction from a depth camera, Consumer Depth Cameras for Computer Vision – Research Topics and Applications, London, str. 71–98, 2012.

[42] (2016) C++ implementacija detekcije človeškega telesa za Microsoft Kinect napravo. Dostopno na: https://github.com/joaquimrocha/skeltrack.

[43] (2016) Apple Ipad Mini 2. Dostopno na: http://www.apple.com/ipad-mini-2/specs/.

[44] (2016) Occipital, Structure sensor. Dostopno na: http://structure.io/.

[45] (2016) C++ programski jezik. Dostopno na: http://www.cplusplus.com/l.

[46] (2016) Objective C programski jezik. Dostopno na: https://developer.apple.com/library/mac/documentation/cocoa/conceptual/programmingwithobjectivec/introduction/introduction.html.

[47] (2016) OpenCV knjižnica. Dostopno na: http://opencv.org/.

[48] P. Viola, M. J. Jones, Robust real-time face detection, International journal of computer vision, 57 (2), str. 137–154, 2004.

[49] P. S. Heckbert, A seed fill algorithm, Glassner, 1 (3), str. 275–277, 1990.

[50] T. Zhang, C. Y. Suen, A fast parallel algorithm for thinning digital patterns, Communications of the ACM, 27 (3), str. 236–239, 1984.

[51] (2016) C++ implementacija Zhang-Suen algoritma. Dostopno na: https://github.com/yati-sagade/zhang-suen-thinning.

[52] P. Rettig, Professional HTML5 mobile game development, pogl. 11, Indianapolis: John Wiley & Sons, 2012.

[53] R. H. Creighton, Unity 3D game development by example: A seat-of-your-pants manual for building fun, groovy little games quickly, pogl. 3, Birmingham: Packt Publishing Ltd, 2010.

[54] J. Makar, Actionscript for multiplayer games and virtual worlds, str. 218–212, Berkeley: New Riders, 2009.